REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2.Nov.00	THESIS

4. TITLE AND SUBTITLE
NEAR REAL TIME ATMOSPHERIC DENSITY MODEL CORRECTION USING SPACE CATALOG DATA

5. FUNDING NUMBERS

6. AUTHOR(S)
1ST LT GRANHOLM GEORGE R

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

8. PERFORMING ORGANIZATION REPORT NUMBER
CY00437

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES
	183
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

CSDL-T-1380

NEAR-REAL TIME ATMOSPHERIC DENSITY MODEL CORRECTION USING SPACE CATALOG DATA

by
George Richard Granholm

June 2000

Master of Science Thesis
Massachusetts Institute of Technology

DRAPER LABORATORY

The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge, Massachusetts 02139-3563

Near-Real Time Atmospheric Density Model Correction Using Space Catalog Data

by

George Richard Granholm

B.S. Astronautical Engineering
United States Air Force Academy, 1998

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2000

Signature of Author: _____
Department of Aeronautics and Astronautics
May 19, 2000

Certified by: _____
Dr. Ronald J. Proulx
Thesis Supervisor, CSDL

Certified by: _____
Dr. Paul J. Cefola
Thesis Supervisor, CSDL
Lecturer, Department of Aeronautics and Astronautics

Accepted by: _____
Nesbitt W. Hagood III
Professor of Aeronautics and Astronautics
Chairman, Department Graduate Committee

[This Page Intentionally Left Blank]

Near-Real Time Atmospheric Density Model Correction Using Space Catalog Data

by

George Richard Granholm

Submitted to the Department of Aeronautics and Astronautics on May 19, 2000 in
Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

Several theories have been presented in regard to creating a neutral density model that is corrected or calibrated in near-real time using data from space catalogs. These theories are usually limited to a small number of frequently tracked "calibration satellites" about which information such as mass and cross-sectional area is known very accurately. This work, however, attempts to validate a methodology by which drag information from all available low-altitude space objects is used to update any given density model on a comprehensive basis. The basic update and prediction algorithms and a technique to estimate true ballistic factors are derived in detail. A full simulation capability is independently verified. The process is initially demonstrated using simulated range, azimuth, and elevation observations so that issues such as required number and types of calibration satellites, density of observations, and susceptibility to atmospheric conditions can be examined. Methods of forecasting the density correction models are also validated under different atmospheric conditions.

Thesis Supervisor: Dr. Ronald J. Proulx
Title: Principle Member of the Technical Staff, The Charles Stark Draper Laboratory

Thesis Supervisor: Dr. Paul J. Cefola
Title: Lecturer, Department of Aeronautics and Astronautics
Program Manager, The Charles Stark Draper Laboratory

[This Page Intentionally Left Blank]

ACKNOWLEDGEMENTS

There are always many people to thank for an endeavor as time-consuming and fraught with challenges as a Master's Thesis, but I have to begin by thanking my two advisors at Draper Laboratory, Dr. Ron Proulx and Dr. Paul Cefola. On countless occasions over the past two years, they have given me advice, encouragement, knowledge, intuition, prodding, and perspective on all areas of my work here at Draper. This thesis truly could not have been even attempted without their help and sponsorship.

I also must express my deepest regards and thanks to my mentors in Russia, Professor Andrey Nazarenko and Dr. Vasiliy Yurasov, for their considerable experience and understanding of atmospheric calibration, and for the countless e-mails which passed to me some small part of that understanding. Their discoveries over the past two decades are the core of the work presented here.

There are a number of other people and departments at Draper who I would like to thank, including: Paul Motyka, Tim Brand, Lee Norris, and Neil Dennehy for their sponsorship of my work and funds for conference travel; George Schmidt and the Education Office for my appointment and for paying the bills; Linda Leonard for accommodating my sometimes unreasonable demands on ELROND and DC1; Dave Carter and Rick Metzinger for their help with GTDS; Barbara Benson for assistance with reams of paperwork; Joe Sartia and others at Repro; Trish and Rosemary in the Travel Office; Ellen, Amy, and the Library staff; and PC Support.

Many members of the MIT community also greatly assisted me over my two years in Cambridge. First of all, I want to give thanks to Dr. Richard Battin for passing on to me a little perspective on the great history and traditions of our field; to Professor Alan Willsky for his furiously-paced lectures and great knowledge of all things stochastic; to Professor John Deyst for his valuable perspective and direction; and to Liz Zotos and Marie Stupard for keeping the Registrar happy throughout.

This work could not have been accomplished without a great deal of help from members of the technical community around the country, and indeed, the world. A lot of the help came from LtCol David Vallado, who provided data, code, advice, and most importantly a sympathetic voice in Colorado. Thanks also must go to Chris Sabol for commiseration and advice on GTDS bugs; to Cheryl Walker and Capt. Darren Roberts for their help in obtaining information on ITT's Modified Atmospheric Density Model (MADM); to Lt. Matt Bradford at NASA-Langley for finding valuable background research; and to Eberhard Gill at ESOC for his help with atmospheric forecasting. I must also thank John Draim of Ellipso Inc. for sponsoring my initial efforts in atmospheric drag analysis.

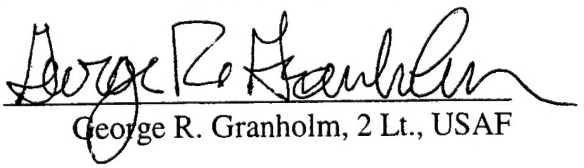
Additional thanks go to Col Kuconis and SSgt Marcaurelle at DET 365 for helping me adjust to life in the real Air Force, and to Malisa Freeland and Capt Melissa Flattery at AFIT for setting up TDYs from Alaska to Florida.

There is a long, distinguished line of Draper Fellows who have worked for Dr. Proulx and Dr. Cefola, most of whom have gone on to very successful careers in the aerospace industry. I have had the privelege of meeting four of them, but at various times I have drawn from the work of Fellows going back to the 1980s. Thanks to all for your careful work and documentation. To Joe Neelon, Brian Kantsiper, and Scott Carter, thanks for showing me there is life after MIT. To my first officemate Jim Smith, you gave me a great example of the right way to do a thesis. Thanks also to my new officemate Raja Chari for your humor and tolerance of my sometimes strange behavior. To Andre Girerd, John Stedl, Paul Goulart, and Andy Grubler, you gave me the opportunity to waste time in new and creative ways, which is at times a necessity.

Many things I will remember most about my years in Boston will be my experiences with my friends and roommates. Thanks to Troy Hacker and Chris Raines for forcing me to actually go out and enjoy myself from time to time. To Jeff Freedman, it's too bad we never got the Hobie-Cat, but I enjoyed all the rollerblading, sailing, and crew practices. To Nick Hague, Scott McKeever, and Jim Wecht, it's always a challenge to find good roommates – but I know now how lucky I was to live with you guys. Good luck in the years to come and I hope we always can keep in touch.

The deepest thanks go to the people without whom I wouldn't be here at all. Thanks Mom, Dad, and Marin for your encouragment and friendly ribbing, and for giving me the motivation and skills to succeed in life. I will be officially entering into the Brugman family in June, but I have felt like you took me in as your son right from the beginning. Thanks to Dennis Brugman for your valuable technical insight and expertise, and to Barbara Brugman for your always thoughtful suggestions and advice. Finally, to the love of my life and my wife-to-be – Rena, I cannot begin to thank you for all the ways you've improved my life. This thesis, and all achievements to come, are dedicated to you.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., with support from Draper Laboratory's DFY99 and 00 IR&D Programs under Astrodynamics IR&D Contract 00-2-837, Project No. 15080.



George R. Granholm, 2 Lt., USAF

ASSIGNMENT

Draper Laboratory Report Number T-1380.

In consideration for the research opportunity and permission to prepare my thesis by and at The Charles Stark Draper Laboratory, Inc., I hereby assign my copyright of the thesis to The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts.

George R. Granholm, 2 Lt., USAF — Date

[This Page Intentionally Left Blank]

TABLE OF CONTENTS

LIST OF FIGURES

[This Page Intentionally Left Blank]

LIST OF TABLES

[This Page Intentionally Left Blank]

Chapter 1 Introduction

1.1 The Problem of Drag Mismodeling

Long-term precision orbit prediction has been of interest since Gauss applied his method of least squares to determine the orbit of the asteroid Ceres in 1801 [2]. Dynamical astronomers in the 19th and 20th centuries grew steadily more proficient at predicting eclipses, the appearance of comets, and the locations of newly-discovered heavenly bodies. However, as the world entered the age of artificial satellites in 1957, precision orbit prediction suddenly became much more important. The unique environment of space was utilized for communications, remote sensing, navigation, and scientific research, and these various types of missions placed new demands on space operations. Today, a wide range of operations depend in varying degrees on the accuracy of perturbation models and propagation methods, including space catalog maintenance, maneuver planning, debris analysis, collision avoidance, and re-entry problems.

Most sources of error in orbit prediction, including a non-spherical Earth, third-body effects, solar radiation pressure, and Earth tides, have been modeled with fair success. Carter showed in 1994 that the Draper R&D Goddard Trajectory Determination System (GTDS) orbit propagation tool is accurate to within one meter of the 1334 km TOPEX reference orbit [45]. However, it has been more difficult to capture the motion of lower-altitude objects due to the inaccuracy of atmospheric density models. Even the models considered to be the closest approximations to real-world conditions, such as Jacchia-Roberts '71 (JR-71) [15] or MSISE-90 [24], can only approach 10% accuracy in quiet conditions and 20-30% in atmospherically perturbed conditions [3]. It is a reflection of the difficulty of density modeling that JR-71 is still one of the most accurate models available even after almost thirty years of research [4].

This is not to say that our understanding of the atmosphere has not grown since 1971. The past decade in particular has seen attempts to produce entirely new density models based on physical principles rather than empirical observations. This thesis is not

such an attempt. Instead, the goal is to outline methods by which the accuracy of already-existing density models may be improved using information currently available from catalogues of frequently tracked space objects. The methodology is developed with sufficient generality such that it may be applied to any thermospheric model that can demonstrate a reasonable level of accuracy over the long term

1.1.1 Overview of Atmospheric Structure and Models

Before specific models of the upper atmosphere are discussed in detail, it is important to provide some explanation of relevant terms and underlying principles. The atmosphere of the Earth can be divided into distinct regions according to several criteria, but the most common criterion is temperature.

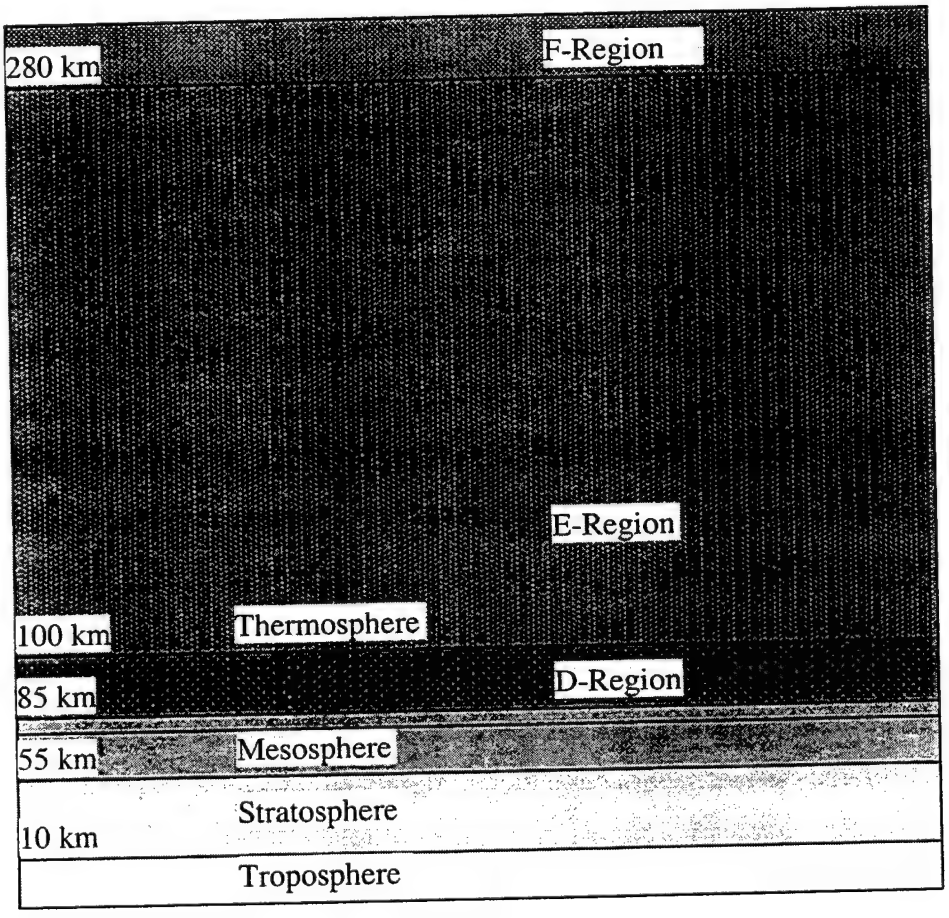


Figure 1.1: Layers of the Amosphere and Ionosphere

The lowest ten to twenty kilometers of the atmosphere is referred to as the *troposphere*, literally the "region of change." The troposphere is characterized by decreasing temperature and terminates at the *tropopause*. Above the tropopause is the *stratosphere*, in which temperature generally increases until about 50-55 kilometers. The upper boundary of this inversion layer is the *stratopause*, which is followed by the *mesosphere*. The mesosphere features steadily decreasing temperature up to the *mesopause*, at 80-85 kilometers. The region above the mesopause is the *thermosphere*. This is the region most frequently modeled for purposes of spacecraft drag [18].

Some other layers of the atmosphere relevant to density modeling are the *ionosphere*, *homosphere, heterosphere*, and *exosphere*. The *ionosphere* is the layer starting at 70-80 kilometers in which ionization of one of the atmospheric constituents is significant. The ionosphere can be divided into a number of regions according to electron density, beginning with the D-region under 100 km, continuing through the E-region, and topped by the F-region above approximately 280 km [31].

The *homosphere* extends from the surface to approximately 90 kilometers. It is characterized by constant atmospheric composition as measured by the mean molecular mass. Dissociation of molecular oxygen leads to decreasing mean molecular mass above the *homopause* and into the *heterosphere* [18]. The heterosphere is characterized by diffusive equilibrium of its major constituents, which include molecular nitrogen (N_2), atomic oxygen (O), helium (He), and atomic hydrogen (H) [30]. Each species is distributed in altitude independently of the molecular weight of other species. Because the scale height of each constituent is inversely proportional to molecular mass, the bottom layer of the atmosphere is primarily molecular nitrogen. Above this layer is a layer of predominantly atomic oxygen at space shuttle altitudes, a layer of helium, and a top neutral layer of atomic hydrogen [30]. High-energy solar rays also produce ions of each of these species, but only the O^+ ion contributes to drag in a significant manner [30]. The hydrogen layer lies primarily in the *exosphere*, which is the highest region of the atmosphere and is where atmospheric particles can move in free orbits, subject only to gravitational forces.

These layers of the atmosphere are by no means axially symmetric or static over time, as was once thought. The most significant spatial and temporal variations can be grouped into categories as follows:

(1) The density of each constituent is directly proportional to its temperature, and the primary source of atmospheric heat is extreme ultraviolet (EUV) solar radiation. EUV radiation exhibits long-term variations over the course of the 11-year solar cycle, and short-term variations related to active regions on the solar surface which appear once per solar rotation (~27 days) [5].

(2) The atmosphere rotates approximately with the Earth, meaning that temperature also varies as a function of solar hour angle, reaching a maximum at 2:00 PM local time and a minumum at 3:00 AM local time. This variation is known as the *diurnal* (day-night) effect. Also grouped with the diurnal variation are latitudinal dependencies. The maximum density was once thought to follow the declination of the sun, but is now known to occur at approximately twice the latitude of the sub-solar point [28].

(3) Researchers in geodesy have observed a *semiannual* variation in thermospheric density, leading to density maxima in April and October and minima in January and July [6]. The physics causing this phenomenon are not fully understood, but many researchers believe the effect is caused by seasonal-latitudinal variations in the mesosphere [28]. Another hypothesis explains the semiannual effect as the result of seasonally varying interaction of the solar wind with the magnetosphere, caused by inclination of the dipole with respect to the ecliptic plane. Regardless of cause, the year-to-year semiannual variations are irregular and defy precise modeling or prediction [32].

(4) Coronal Mass Ejections and other solar eruptions deposit high-energy plasma into the solar wind, which in turn injects energy into the magnetosphere in the form of Joule heating [7]. The resulting geomagnetic storms are the single largest factor affecting short-term fluctuations in thermospheric density [5].

(5) Another variation described by Keating et. al. [30] is the "winter helium bulge," which refers to increased concentrations of helium in the winter hemisphere (on the other side of the equator from the sub-solar point). There is also a corresponding increase of positively charged atomic oxygen (O^+) in the summer hemisphere.

Most thermospheric models have attempted to describe some or all of these variations, with varying success. Jacchia's 1971 model [15] with Roberts' analytical evaluation methods [16] is based on empirical fitting of total density and incorporates all of the above variations. JR-71 has been shown to display a good combination of speed and accuracy [33]. In addition, JR-71 is in wide use today, and will therefore be the truth model of choice for the numerical simulations in this thesis. However, it is instructive to briefly review major efforts in density modeling since 1971. A detailed but by no means complete list of thermospheric models developed in the last thirty years would include:

- Jacchia '77 [17]
- Barlier's DTM 78 model [19]
- the 1979 Aeros model of Köhnlein [20]
- the NASA/MSFC GRAM model [46]
- Alycadé's 1981 model [21]
- the Russian GOST model [47]
- Hedin's MSIS models [22-24]
- the MET model developed by Hickey in 1988 [25]
- Sehnal and Pospisilova's TD 88 model [26]
- the Fuller-Rowell CTIM model of 1996 [27]
- Hicks' 1997 GWUAM model [28]
- Owens' development of MET-99 last year [29]

The sheer number of different models should indicate that the problem of density modeling continues to merit serious attention.

1.1.2 Some Problems with Current Thermospheric Models

Why such difficulty in modeling and predicting the effects of drag on a spacecraft? To more effectively address this question, we may categorize problems with density modeling into three areas: (1) thermospheric density model limitations; (2) data prediction errors; and (3) errors in modeling the drag force exerted on a spatially and compositionally complex spacecraft. This work focuses on improving results in the first two areas.

(1) Thermospheric density model limitations. Thermospheric models vary in complexity, size, and accuracy, but most models share certain characteristics and assumptions. All models recognize the influence of EUV radiation on density. However, direct measurements of EUV radiation are not generally available, so we are forced to rely on the ground-based 10.7-cm solar flux (usually designated as $F_{10.7}$) as a proxy measurement. Marcos showed in 1997 that the $F_{10.7}$ measurements do not completely represent the actual EUV radiation, and can result in amplitude errors under spectral analysis of up to 30% [6].

We can measure the effect of the solar wind on the Earth's magnetic field with the Kp index, a globally averaged logarithmic index of geomagnetic activity. Some models also use geomagnetic information in the form of the daily Ap index, which is a linearized version of the sum of Kp values averaged over one day. The Ap index, while able to capture the average effect of geomagnetic storms on thermospheric density, cannot accurately specify the spatial distribution and temporal evolution of the magnetospheric sources of energy. In addition, no widely available thermospheric model has incorporated the tidal and wave motions of the lower and middle atmosphere. The result of these limitations is that models are limited to 10% accuracy even when all necessary data are available.

(2) Data prediction errors. When dealing with prediction problems where future $F_{10.7}$ and Ap values are not available, we must use some kind of forecasting algorithm or rely only on past data. Most models use a combination of daily

$F_{10.7}$ values and 81-day averages of $F_{10.7}$ (denoted by $\overline{F}_{10.7}$) centered on the epoch of interest to calculate the exospheric temperature, which is usually the basis for the neutral thermospheric density. The daily $F_{10.7}$ values fluctuate in accordance with the appearance of active regions on the solar surface over the course of its 27-day rotation. Nostrand has shown that the Air Force Global Weather Central (AFGWC) can only accurately predict $F_{10.7}$ up to three days [53]. When the necessary data are not available, many agencies (including the Air Force Space Warfare Center) use a running average taken over the past 90 days as an estimate of $\overline{F}_{10.7}$. If the solar flux does not conform to its past behavior, this estimation can result in errors of up to 40 solar flux units, or equivalently a density difference of about a factor of two [8]. In addition, the long-term Ap trend is dictated by the 11-year solar cycle and can be predicted with reasonable accuracy, but the day-to-day Ap measurements fluctuate in a semi-random manner. Therefore, the predicted values of Ap and $\overline{F}_{10.7}$ can at best capture long-term trends and will not reflect day-to-day fluctuations.

(3) Errors in drag force modeling. Most current thermospheric models do not incorporate detailed information about gas-surface interaction between air molecules and three-dimensional spacecraft surfaces [9]. Instead, all information about how a spacecraft interacts with the thermosphere is reduced to one dimension in the form of the unitless drag coefficient, C_D. This coefficient is usually assumed to be constant and is used as a parameter by which the estimation process can adjust the effects of drag to best fit observations. Unless the attitude, shape, and composition of a space object is known very accurately, the best estimate of the coefficient of drag (on which the ballistic factor directly depends) may deviate from the true, time-varying coefficient by as much as 15% [4].

1.2 The Density Correction Process

It seems clear that to more effectively model short-term variations in density, we must come up with a way to more frequently and comprehensively measure changes in

atmospheric conditions. Several investigators have suggested the launch of low-cost "calibration" satellites to directly measure atmospheric density in the low-altitude regime; Laneve [48] proposes a small constellation of spherical satellites in highly elliptical orbits. However, an even lower-cost approach is to use data that we currently have in the US Space Catalog to estimate corrections to existing atmospheric models in near-real time. The basic idea is that if we know the orientation, mass, and cross-sectional area of a particular spacecraft, but our differential corrections process is telling us something different, the difference between the truth and estimate is a reflection of the inaccuracy of the density model for that particular range of time and altitude. This assumes that we can isolate the effect of drag perturbations on the objects from the effects of other perturbations. If we take enough measurements of diverse low-altitude objects over a long enough period of time, we should be able to form a density correction model on a global scale. The density correction model may also be forecast into the future for orbit prediction purposes. Not all objects used in the correction model need to have precisely known cross-sectional area and mass. In fact, the process may be used to better estimate unknown spacecraft characteristics as a by-product of density correction.

1.2.1 Basic Operation

We initially select 200-300 frequently tracked objects that have sufficiently diverse inclinations, eccentricities, and perigee heights between 200 and 600 km. We then designate objects that have constant and precisely known ballistic factors as "standard" satellites. The ballistic factor is defined as

$$k_i = \left(\frac{C_D A_x}{2m} \right)_i \tag{1.1}$$

where C_D is the coefficient of drag, A_x is the cross-sectional area perpendicular to the satellite's motion, and m is the mass. The subscript i refers here to the i^{th} satellite, where the satellites are numbered from 1 to n.

An example of a standard satellite might be a spherical object with known mass and composition in a stable, near-circular orbit. The drag coefficient for these types of satellites does not vary appreciably over time or altitude, as long as the satellite remains

in the low-altitude drag regime (under 600 km) [4]. Because we can essentially eliminate errors due to unknown mass, cross-sectional area, or drag coefficient, the standard satellites will form the core of our density correction database. Nazarenko has shown that the density correction process can be effective if 1/10 objects in the database are standard [1].

The remaining objects are designated as "non-standard" satellites. We define a constant "true" ballistic factor for these objects as well, although the actual ballistic factor may be changing from one moment to the next. We also know the degree of variability for each ballistic factor in the form of the standard deviation.

For standard satellites, the values C_D, A_x, and m are well known and k_i can be calculated to a high level of accuracy. For non-standard satellites, the true ballistic factors can be approximated by averaging ballistic factors taken over a sufficient number of preceding solar rotations. We will not be able to gain as much information about the inaccuracy of our density model from the non-standard satellites, but some error can be removed using measurements from the standard satellites as calibration data. This process and more details on the estimation of "true" ballistic factors for non-standard satellites are presented in Chapter Two.

Using precise orbit propagation tools (i.e. special perturbations or semi-analytic theory) and a sliding three-day window of observations, we estimate the orbits and ballistic factors of all satellites in the database. The deviations from the true ballistic factors should reflect the amount of error in the given thermospheric model for a particular altitude and time. We "attribute" each observed ballistic factor to a specific time and altitude, group ballistic factors into 3-4 hour spans, and construct linear models of density variations for each span. The linear models will be piecewise constant for each span, and will depend only on altitude. By using 200-300 satellites in the database, we ensure that each three-hour span will contain at least 35-40 ballistic factor estimations. According to the work of both Storz [11] and Nazarenko [1], this seems to be the minimum number of measurements necessary for obtaining adequate global coverage of the thermosphere. Over the time period of interest, our thermospheric model has now been calibrated to data taken from the space catalog. This process should account for the

major errors caused by limitations in the model as discussed in area (1) under Section 1.1.2 above.

Once the density variation models have been calculated, the original thermospheric model plus the corrections can be used to estimate the orbit and ballistic factor of a new, possibly unknown space object. The orbital elements should be more accurate as we are no longer trying to fit observations to an incorrect model. Also, because we have accounted for the limitations of the density model, we are left only with errors in the ballistic coefficient. We should therefore be able to gain more information about the "true" ballistic coefficient of the target object and possibly information about coefficient decay rates, mass, or shape as well.

To reduce error associated with Section 1.1.2 (2), data prediction, we can treat the density variation model coefficients as observations of stochastic processes. The deterministic components are estimated first, and a Kalman filter is then used to forecast the random component. It is thus possible to obtain density variation models for time periods during which we have no measured thermospheric data. This approach is in effect a way of forecasting values of Ap and $F_{10.7}$ in a stochastically optimal way.

Regardless of whether real observations are available, it will be necessary to construct an observation simulator. This is so that we can completely validate the algorithms before moving on to real data. First, the ballistic factors and mean elements of the standard and non-standard satellites are generated with sufficient diversity in orbital eccentricity, perigee height, and inclination. We simulate truth orbits of all satellites using a high-precision propagator, the truth density model, and the true ballistic factors. The orbits are simulated for a relatively long period of time (at least 20-30 days) so that we can gain some sense of how the process performs over changing atmospheric conditions. This long time period is also necessary to allow the estimation of "true" ballistic factors for non-standard satellites.

The positions and velocities of the satellites are used as inputs to the observation simulator, and the output observations are in the form of time-tagged range, azimuth, and elevation with appropriate noise characteristics. The ballistic factors of the non-standard satellites are then reset to a-priori values, and we fit the observations for each satellite using a sliding three-day window as described above. The fit model will be either a

simpler thermospheric model or the truth model but with smoothed Ap and $F_{10.7}$ inputs. The ballistic factor estimations are used to construct the 3-hour density variation models over the entire 20-30 day time period. If the process works, the simpler fit model plus the corrections should equal the truth model. For testing, a target orbit and target ballistic factor are estimated using the corrected density model, and accuracy of the density variation prediction algorithms is investigated by comparing the estimated orbit with the "truth" orbit.

1.3 Previous Work

The idea of using observations of drag-perturbed spacecraft to improve thermospheric models is not a new one. In fact, the initial derivation of most models was done in precisely such a manner, primarily because the atmosphere is not understood well enough to construct a model based solely on physical principles. However, the particular method of model correction has varied widely. One idea presented by Barker et. al. in 1989 [34], specifically applied to the decay problem, is to parameterize the ballistic coefficient as a linear function of time. The rate of change of ballistic coefficient is then solved for in a differential correction process and used for orbit prediction. Wright presented a more comprehensive methodology in 1990 [35], in which a subset of atmospheric density parameters is estimated for each spacecraft in a sequential filter. These parameters include coefficients that appear in Jacchia's empirical equations for exospheric temperature, diurnal variations, geomagnetic disturbances, and other effects. The parameters are then processed by a second sequential filter to provide density corrections in near-real time.

More recently, Marcos et. al. [6] presented a scheme for correcting a given atmospheric model using observations from a calibration satellite, where the satellite's true ballistic factor is known very precisely. These observations are fit over a few days and the resulting ratio of "adjusted" to true ballistic coefficient is used to correct the model on a global scale. Storz [11] has also outlined a somewhat different approach in the past year, in which estimated ballistic factors are used to solve for energy dissipation rates for a number of calibration satellites. The energy dissipation rates are then used to

estimate coefficients for a spherical harmonic expansion of exospheric temperature, which is one of the primary inputs into Jacchia's models.

Some of the most comprehensive work in dynamic atmospheric correction, however, has been performed over the past decade by A.I. Nazarenko and V.Yurasov [1, 12, 13]. This thesis extends and applies some of their methods in a more operational environment. The underlying mathematical theory is refined and presented in more detail. The theory is independently verified with a simulation of all phases of the problem, including data generation and differential corrections. The dependence of orbit determination accuracy on atmospheric conditions, quantity, and quality of observations is investigated. Finally, we will verify the effectiveness of "true" ballistic factor estimation and density variation forecasting.

1.4 Outline of Thesis

The next chapter of the thesis will explain the density correction process in more mathematical detail, including the calculation of density variations, estimation of "true" ballistic factors, and forecasting of density variations. The third chapter outlines the tools and software used in this thesis, including the Goddard Trajectory Determination System (GTDS) and the density correction software written by the author. The fourth chapter describes the setup of the simulation and types of test cases to be executed. Chapter Five discusses the results of the test cases, and Chapter Six will present conclusions and future work. Appendix A will include additional results in graphical form, and Appendix B will discuss issues dealing with operation of GTDS on the UNIX system.

Chapter 2 Mathematical Specifications

Most of the derivations in this chapter follow the general flow of the reports authored by A. I. Nazarenko and commissioned by Draper Laboratory over the past three years [1]. However, an attempt has been made to correct a few errors and standardize the notation. Additionally, the sections on estimation of "true" ballistic factors and forecasting of density variations are presented in more detail. Systematic derivations allow a reader not already familiar with some of the concepts to follow along without too much difficulty.

Presented below is a flow diagram of each stage in the density correction process. The diagram assumes that the observations are simulated. Each stage in density correction and prediction is derived in the sections to follow.

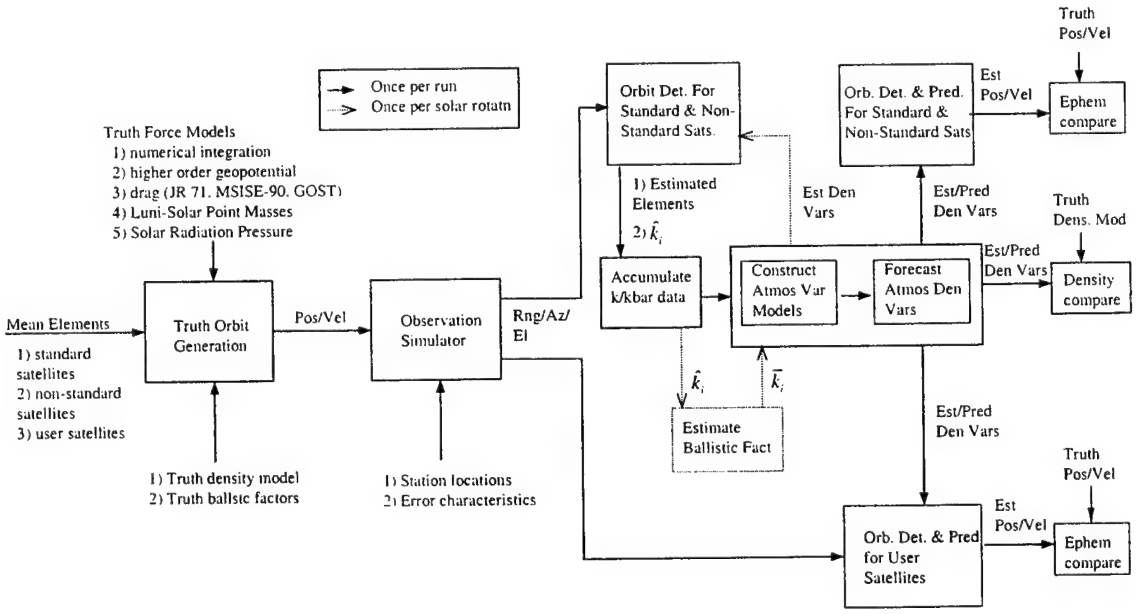


Figure 2.1: Density Correction Flow Diagram Using Simulated Observations

2.1 Construction of Density Variations

There are two approaches that one can take in obtaining the ballistic factor estimations from the observation data: batch-fit methods, and recursive filtering

techniques. An overview of the basic concepts for each of these techniques is discussed below.

2.1.1 Batch-Fit Concepts

We are initially given observations for a large group of "standard" and "non-standard" satellites with perigee heights between 200 km and 600 km. The observations are distributed over at least an interval of one solar rotation (≈27 days) to have enough data to come up with initial estimates of non-standard satellite ballistic factors. We are also given an a-priori density model, which for simulated observations can be the same overall model as the truth model but without short-period density fluctuations. The a-priori model can also be an entirely different model than the truth. If the observations are not simulated, we will use the best available model including all long and short-period perturbations. Observations taken an average of three times a day from somewhere in the range of 200-300 satellites should provide enough information for problem solution. The exact number and duration of needed observing passes is unknown, and can be adjusted as necessary.

We want to use these observations to estimate ballistic factors, which are then used to construct linear density variation models; however, we must first "localize" the ballistic factor estimations with respect to altitude and time. Beginning with the first three days of data, we estimate the orbital elements and ballistic factors for each of our n space objects. The estimated ballistic factor for each satellite is in some sense a reflection of the average drag that the satellite experienced over the fit span; therefore, each ballistic factor can be localized to the height of perigee. The error associated with this attribution should be insignificant because atmospheric density decreases exponentially with increasing altitude. The time attributed to each density variation depends on the estimation method. For batch least-squares fitting, the estimator attempts to find a value of the ballistic factor such that the residual errors over the entire interval are minimized. Therefore, assuming the observations are distributed somewhat uniformly across the fit span, the estimated ballistic factor should be attributed to the midpoint of the observations for that particular satellite. Once we have estimated and localized all of the n ballistic factors, the three-day fit window is shifted forward by three

hours, and we re-estimate orbital elements and ballistic factors for all satellites for which we have obtained new observations. If observations for each object come in every 8 hours, the average number of new ballistic factor estimations obtained every three hours will be equal to $3n/8$. This means that n should be ≥ 214 to ensure an average of 80 new ballistic factor estimations per three hours.

We are left with a large number of ballistic factor estimations, each one of which is associated with a particular satellite, altitude, and time. We now need to group the estimations into j spans of 3-4 hours each so that we may construct a linear density variation model for each span. The length τ_j of each span, where j ranges from 1 to N, should be minimized so that the density variation model can capture short-period fluctuations in density. However, τ_j must be long enough to contain a sufficient number of ballistic factor estimations, as was discussed previously. We will set τ_{min} equal to 3 hours. The first density variation model span will begin about 1.5 hours before the midpoint of the first three-day fit window, since that is where the first ballistic factor estimations will be attributed. We assign the beginning of the first span to T_1, and count the number of estimations contained in $[T_1, T_1 + 3\text{hrs})$, eliminating any estimations from identical satellites. If the number of estimations is greater than 35, we assign $T_2 = (T_1 + 3$ hrs) and continue with the second span. Otherwise, we extend the first span by 5-minute intervals until we have enough estimations. We continue with the process until the start and end times of all N spans have been defined. We will now adopt the following notation: $\hat{k}_{ij}$ refers to the ballistic factor estimation associated with the i^{th} satellite and the j^{th} span; h_{ij} is the height of perigee attributed to $\hat{k}_{ij}$; and t_{ij} is the time attributed to $\hat{k}_{ij}$. We also have σ_i^2, which is a measure of the variability of the ballistic factors for satellite i, and which will be further defined below.

2.1.2 Recursive Filtering Concepts

Most of the concepts described for batch-fit techniques also apply to recursive filtering methods, but there are a few important differences relating to the attribution of ballistic factors. We begin with the same number and frequency of observations, but it may be necessary to use more than three days of data to allow the filter to converge to a

valid solution. Once solutions have been obtained for all n satellites, we assign the current time to the beginning of the first density variation model span (T_1), and take additional observations for three hours. Each observation is used to recursively estimate the real-time orbital elements and ballistic factor, meaning that the time attribution for each ballistic factor, t_{ij}, is simply equal to the time of estimation. The attribution altitude remains the height of perigee, also at the time of estimation (if the height of perigee is changing over time). If there are greater than 35 estimations after 3 hours, we assign $T_2 = T_1 + 3$ hrs and move to the next span; otherwise, we take more range/az/el observations until we obtain 35 estimations. As soon as we reach the end of the available observations, our sorting process is finished, and we are ready to construct our linear density variation models for each span.

2.1.3 Relating Ballistic Factors to Density Variations

We next will derive the relationship between the ratio of estimated to "true" ballistic factor and the ratio of the difference in true and modeled density to the modeled density. The true density at a given time and altitude is related to the modeled density by the following equation:

$$\rho = \rho_m + \delta\rho = \rho_m\left(1 + \frac{\delta\rho}{\rho_m}\right) \tag{2.1}$$

where ρ_m refers to the density calculated by the model. We are trying to measure the density variation term $\delta\rho / \rho_m$ at a particular altitude and time given the true ballistic factor k_i and the estimated ballistic factor $\hat{k}_{ij}$, which is obtained from the observations as described above. We can choose an orbital element directly related to energy of the orbit (such as period or semi-major axis) and write its true rate of change over one revolution as the product of the true ballistic factor, true perigee density, and some function of the orbital elements. For the purposes of our simulation, we will choose the period rate, denoted by $\dot{T}_i$:

$$\dot{T}_i = k_i \cdot \rho(h_{ij}, t_{ij}) \cdot f(\underline{x}) \tag{2.2}$$

The term $\underline{x}$ refers to the state vector, which may be any complete set of the six orbital elements. We estimate the period rate from our observations, and use the ballistic factor to fit the observation to the modeled density at perigee:

$$\hat{\dot{T}}_{ij} = \hat{k}_{ij} \cdot \rho_m(h_{ij}, t_{ij}) \cdot f(\underline{x}) \tag{2.3}$$

This equation assumes that our perturbation model can very accurately describe other perturbations that affect the energy of the orbit, such as third-body perturbations, solar radiation pressure, and non-spherical Earth effects. In other words, it is necessary to use semi-analytic techniques or special perturbations to propagate orbits. Otherwise, we will not definitively know whether our estimator is adjusting the ballistic factor in response to errors in the atmospheric model or in response to other model errors. We define the relative error ε between the real and estimated period rate with the following equation:

$$\dot{T}_i = \hat{\dot{T}}_{ij}(1+\varepsilon) \tag{2.4}$$

Combining Eqs. (2.2)-(2.4), we can write

$$\frac{\hat{k}_{ij}}{k_i} - 1 = \frac{\rho(h_{ij}, t_{ij})}{\rho_m(h_{ij}, t_{ij})} - 1 - \left(\frac{\hat{k}_{ij}}{k_i}\varepsilon\right) = \frac{\delta\rho}{\rho_m}(h_{ij}, t_{ij}) - \left(\frac{\hat{k}_{ij}}{k_i}\varepsilon\right) \tag{2.5}$$

If we assume that the observed period rate is a good approximation of the true period rate, i.e. $\varepsilon = 0$, we now have a relation between the density variation at perigee and the ratio of estimated to true ballistic factors. However, unless we are dealing with a standard satellite, we do not know the true ballistic factor k_i. For these non-standard satellites, we can use the time-averaged ballistic factor $\bar{k}_i$ as an approximation of the true ballistic factor; methods for obtaining $\bar{k}_i$ will be discussed in Section 2.2. These assumptions and the previous derivations result in the fundamental equation for constructing density variations

$$\frac{\delta\rho}{\rho_m}(h_{ij},t_{ij}) = \left(\frac{\hat{k}_{ij}}{\bar{k}_i}\right) - 1 \tag{2.6}$$

where the subscripts i and j denote the i^{th} satellite and j^{th} span, as was mentioned before. Each density variation is used to construct the time and altitude-dependent density variation model for each span.

2.1.4 Least-Squares Solution

At this stage of the problem, we have at least 35 density variation measurements for each of the N three-four hour time spans in the form of the ratio of estimated to "true" ballistic factors. The satellites used in any particular span are a subset of the entire set of satellites, which we will denote by $I = \{1,\ldots,n\}$. The subset of satellites for span j, which we denote by $n_j \subset I$, will change from one span to the next. Conversely, the spans that satellite i appears in is a subset of the entire set of spans, which we will denote by $J = \{1,\ldots,N\}$. The subset of spans for satellite i, which we denote by $N_i \subset J$, is different for each satellite.

Our task is to now use the density variation measurements to construct piecewise-constant linear models, which are assumed to take the following form:

$$\frac{\delta\rho}{\rho_m}(h_{ij},t_{ij}) = \left(\frac{\hat{k}_{ij}}{\bar{k}_i}\right) - 1 = b_{1j}f_1(h_{ij}) + b_{2j}f_2(h_{ij}) + \Delta_{ij} \tag{2.7}$$

By "piecewise-constant," we mean that each linear model is constant with respect to time over its particular span j. The term Δ_{ij} is the residual error for each estimation and is assumed to be zero-mean white Gaussian noise (WGN) with variance σ_i^2. The linear function are defined as $f_1(h_{ij}) = 1$ and $f_2(h_{ij}) = (h_{ij} - 400)/200$, the forms of which impart a physical meaning to the coefficients b_{1j} and b_{2j}: the first measures the relative variation of density at 400 km, and the second characterizes the change of relative variations within ± 200 km of the mean altitude of 400 km. Although the density models are specifically tailored to the 200-600 km range, it should be possible to apply the equations above 600 km without negative consequences. This is because it is very likely

that lower-altitude error trends will continue in some fashion at higher altitudes, and also because drag effects are significantly reduced at these altitudes.

The reader should note that the linear correction models do not depend on longitude or latitude, meaning that the same correction is applied at a particular altitude and time regardless of position. This feature could lead to errors when the atmosphere is perturbed by location-dependent sources, such as geomagnetic disturbances. However, the simplicity of the correction equations was found by Nazarenko to be a necessary limitation of density correction. A possible consequence of increasing the number of terms in the correction equations is that the useful information contained in the observation data must be spread over a greater number of coefficients, resulting in lower accuracy for all [12]. It may be feasible to construct more complex models with greater quantities and quality of observations.

The linearity of the functions allows for the direct solution of the problem using analytical least-squares techniques. To properly justify the use of least-squares for this problem, we should demonstrate that the assumption of Gaussian white noise is a good one. Implicit in this assumption is that the average error in our given atmospheric model is also Gaussian. Jaeck-Berger and Barlier obtained approximately 12000 measurements of density from 80 satellites over a three-year period, and calculated the ratio of true density to the density calculated by the Jacchia '71 model [36]. Their results are shown in the figure below:

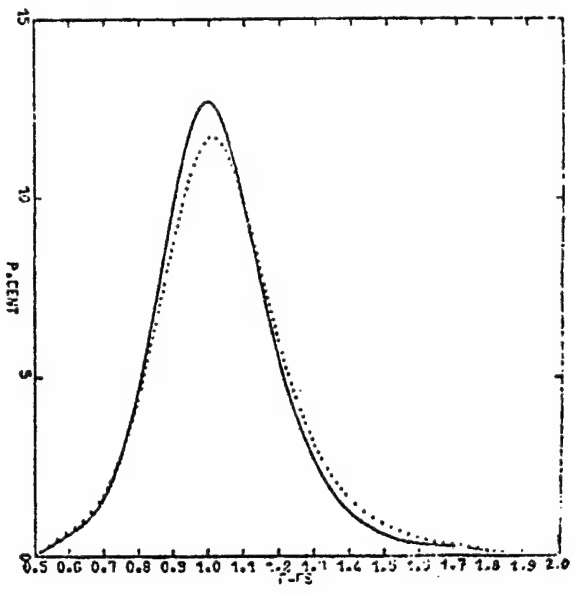


Figure 2.2: Ratio of True Density to Jacchia '71 Density [36]

The dashed line refers to the actual ratio of true density to J71 density, while the solid line is the result of an empirically corrected J71 model devised by Jaeck-Berger and Barlier. Corrected or uncorrected, however, the errors are seen to be distributed in a Gaussian fashion with mean of approximately one.

We are now ready to define our cost function and minimize with respect to the model coefficients, but we first will redefine several quantities in vector notation. The residual error vector is defined by:

$$\boldsymbol{\Delta}_j^T = [\Delta_{1j} \quad \cdots \quad \Delta_{n_j j}]^T \tag{2.8}$$

where each residual error is found from the following equation:

$$\Delta_{ij} = \left(\frac{\hat{k}_{ij}}{k_i} \right) - 1 - \sum_{q=1}^{2} b_{qj} \cdot f_q(h_{ij}) \tag{2.9}$$

The vector defined in Eq. (2.8) does not really contain the residuals for the satellites $i = 1,\ldots,n_j$, but in actuality is defined by the ordered subset $n_j \subset I$. However, from this point forward, we will abuse notation in a similar manner for all vectors

composed of elements taken from the subsets n_j or N_i. The weighting matrix $\mathbf{P_j}$ is a diagonal n_j x n_j matrix with the weight for the i^{th} satellite equal to $1/\sigma_i^2$. In reality, we will not know the "true" variance, but we can use the time-averaged estimate $\overline{\sigma_i^2}$ instead. Our cost function can now be simply written as

$$I(\mathbf{b_j}) = \mathbf{\Delta_j^T P_j \Delta_j} \tag{2.10}$$

where $\mathbf{b_j^T} = [b_{1j}\ b_{2j}]^T$. If we define the vector $\mathbf{a_j}$ as

$$\mathbf{a_j^T} = \left[\left(\frac{\hat{k}_{1j}}{\overline{k_1}} \right) - 1 \quad \cdots \quad \left(\frac{\hat{k}_{n_j j}}{\overline{k_{n_j}}} \right) - 1 \right]^T \tag{2.11}$$

and the n_j x 2 matrix $\mathbf{F_j}$ as

$$\mathbf{F_j} = \begin{bmatrix} f_1(h_{1j}) & f_2(h_{1j}) \\ \vdots & \vdots \\ f_1(h_{n_j j}) & f_2(h_{n_j j}) \end{bmatrix} \tag{2.12}$$

then the cost function becomes

$$I(\mathbf{b_j}) = (\mathbf{a_j} - \mathbf{F_j b_j})^T \mathbf{P_j} (\mathbf{a_j} - \mathbf{F_j b_j}) \tag{2.13}$$

Taking the first derivative with respect to $\mathbf{b_j}$, setting the equation equal to zero, and solving, we obtain the traditional least-squares solution:

$$\hat{\mathbf{b}}_\mathbf{j} = (\mathbf{F_j^T P_j F_j})^{-1} \mathbf{F_j^T P_j a_j} \tag{2.14}$$

2.1.5 Validation of Solutions

The solutions obtained from Eq. (2.14) above must be checked for physical authenticity. We set the following constraints on allowable values of density variations:

$$\left|\frac{\delta\rho}{\rho_m}(h_{ij}=200)\right| = \left|b_{2j} - b_{1j}\right| < 0.3 \tag{2.15}$$

$$-0.5 < \frac{\delta\rho}{\rho_m}(h_{ij}=600) = b_{1j} + b_{2j} < 2.0 \tag{2.16}$$

These bounds are commensurate with the max 30% error in most density models at low altitudes, and with the greater errors occurring at higher altitudes. Nazarenko [1] found that almost all density variations fell within these boundary values, but the values may be adjusted as necessary.

2.2 Estimation of Ballistic Factors

Through all previous derivations, we have assumed that we accurately know the "true" ballistic factor k_i and its variability in the form of σ_i^2 for each of the n satellites. However, this is usually not the case for non-standard satellites, which will comprise a majority of the space objects used in the atmosphere correction service. We instead must use some kind of estimation process to find time averaged versions of these quantities, which are expressed as $\overline{k_i}$ and $\overline{\sigma_i^2}$. The fundamental assumption on which the rest of the derivations will rely is that the average residual error for a particular satellite is equal to zero:

$$\mathrm{E}[\Delta_i] = 0 \tag{2.17}$$

Here, E is the expectation operator and the terms written in the Arial font are considered random variables. Note that the residual term here refers to residual errors averaged across many spans for one satellite, where the residual vector in Eq. (2.8) refers to a particular span j but many different satellites. This equation will be valid if the differences between the true and modeled density can be accurately captured by the density variation models defined in Eq. (2.7). This in turn means that on the average, the modeled density should be an accurate representation of the true density, as shown in Figure 2.2 above.

The first task is to define a measure of quality for the density variation models. A natural measure is the averaged sum of residuals for a particular satellite over the appropriate spans. For standard satellites, the measure is defined as

$$Q_e = \frac{\sum_{j \in N_e} \Delta_{ej}}{|N_e|} \tag{2.18}$$

where $e \in I_e$, $I_e \subset I$ is the subset of standards satellites, and $N_e \subset J$ is the subset of spans which contain ballistic factor estimations from standard satellite e. The term $|N_e|$ refers to the ordinality (i.e. size) of the subset N_e. A similar measure for non-standard satellites can be defined with

$$Q_i = \frac{\sum_{j \in N_i} \Delta_{ij}}{|N_i|} \tag{2.19}$$

where $i \in I_f$, $I_f \subset I$ is the subset of non-standard satellites, and $N_i \subset J$ is the subset of spans which contain ballistic factor estimations from non-standard satellite i. Note that $I_f \cup I_e = I$.

The method of estimation depends on the number of standard satellites available. The equations are first derived using one standard satellite, and are then generalized for the case in which we have multiple standard satellites.

2.2.1 Ballistic Factor Estimation with One Standard Satellite

We seek to find the set of "true" ballistic factors that minimize the quality measures defined in Eqs. (2.18) and (2.19). We will first make use of the information provided by the standard satellite in the form of Q_e. It is reasonably certain that changes in the ballistic factor of the standard satellite are due to density model inaccuracies and not to changes in satellite attitude or composition. Therefore, if Q_e is not equal to zero, its magnitude is in some sense a measure of the validity or "bias" of the density variation models as a whole.

As discussed above, the residuals for any satellite are expected to sum to zero:

$$\sum_{j\in N_i}\frac{\hat{k}_{ij}}{\bar{k}_i}-\sum_{j\in N_i}\left(1+\sum_{q=1}^{2}\hat{b}_{qj}f_q(h_{ij})\right)=0 \tag{2.20}$$

Here, $\hat{b}_{qj}$ is the density variation model coefficient which has been estimated using the uncorrected ballistic factors. The term b_{qj} will refer to the density variation model coefficient calculated using "true" ballistic factors. If we solve for the a-priori ballistic factor $\bar{k}_i$ in the above equation, we find that

$$\bar{k}_i=\frac{\sum_{j\in N_i}\hat{k}_{ij}}{\sum_{j\in N_i}\left(1+\sum_{q=1}^{2}\hat{b}_{qj}f_q(h_{ij})\right)} \tag{2.21}$$

If the true values of the ballistic factors were known and we constructed density variation models using these factors, Eq. (2.20) would remain valid:

$$\sum_{j\in N_i}\frac{\hat{k}_{ij}}{k_i}-\sum_{j\in N_i}\left(1+\sum_{q=1}^{2}b_{qj}f_q(h_{ij})\right)=0 \tag{2.22}$$

Solving for the true ballistic factor k_i,

$$k_i=\frac{\sum_{j\in N_i}\hat{k}_{ij}}{\sum_{j\in N_i}\left(1+\sum_{q=1}^{2}b_{qj}f_q(h_{ij})\right)} \tag{2.23}$$

We now make our key assumption: that each a-priori ballistic factor deviates by a factor ξ from the true ballistic factor. In equation form:

$$\bar{k}_i=\xi\cdot k_i \qquad ,i\in I_f \tag{2.24}$$

Combining Eqs. (2.21), (2.23), and (2.24) leads to

$$\xi \cdot \sum_{j \in N_t} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ij}) \right) = \sum_{j \in N_t} \left(1 + \sum_{q=1}^{2} b_{qj} f_q(h_{ij}) \right) \tag{2.25}$$

We are now ready to solve for ξ using the information obtained from Q_e. Since this information approximately reflects the average or overall "bias" of the variation model due to inaccuracies in non-standard ballistic factors, we can write

$$\xi \cdot \sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ej}) \right) \approx \sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} b_{qj} f_q(h_{ej}) \right) \tag{2.26}$$

Applying the condition of Eq. (2.22) to the standard satellite,

$$\sum_{j \in N_e} \frac{\hat{k}_{ej}}{k_e} - \sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} b_{qj} f_q(h_{ej}) \right) = 0 \tag{2.27}$$

The previous two equations are then combined and the sum of density variation model evaluations is subtracted from both sides to produce

$$\sum_{j \in N_e} \frac{\hat{k}_{ej}}{k_e} - \sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ej}) \right) \approx \sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ej}) \right) \cdot (\xi - 1) \tag{2.28}$$

We observe that the LHS of this equation is equal to $(Q_e \cdot |N_e|)$. Solving for the correction term ξ, the fundamental ballistic factor correction equation is obtained:

$$\xi \approx 1 + \frac{Q_e \cdot |N_e|}{\sum_{j \in N_e} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ej}) \right)} \tag{2.29}$$

This correction factor is applied to each non-standard ballistic factor according to the following equation:

$$\left(\bar{k}_i\right)_{new} = \frac{\left(\bar{k}_i\right)_{old}}{\xi} \qquad , i \in I_f \tag{2.30}$$

The correction process will undoubtedly overestimate some ballistic factors and underestimate others because the correction factor removes the *average* deviation obtained from the standard satellite information. However, the next stage of ballistic factor updating will remove these individual biases using information from each non-standard satellite in the form of Q_i.

We first define a new satellite-dependent correction factor:

$$\bar{k}_i = \psi_i \cdot k_i \tag{2.31}$$

The fundamental equation for the new correction factor is derived using the same methodology as above, except that Q_i is used as the information basis instead of Q_e:

$$\psi_i \approx 1 + \frac{Q_i \cdot |N_i|}{\sum_{j \in N_i} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ij}) \right)} \tag{2.32}$$

A different correction factor is calculated for each satellite and applied using the analogue of Eq. (2.30):

$$\left(\bar{k}_i\right)_{new} = \frac{\left(\bar{k}_i\right)_{old}}{\psi_i} \tag{2.33}$$

It is also necessary to obtain estimates of the variance of residual errors, or σ_i^2. Since the residual error for each satellite and at each span is assumed to be an independent, identically distributed Gaussian random variable, we may use the (efficient) max-likelihood estimate for the variance:

$$\bar{\sigma}_i^2 = \frac{\sum_{j \in N_i} \Delta_{ij}^2}{|N_i|} \tag{2.34}$$

It can be seen from Eqs. (2.29), (2.32), and (2.34) that the updating of ballistic factors depends on the construction of density variation models, and vice-versa. Therefore, iteration will be necessary. To summarize, the density variation models are

first constructed for each time span using the least squares solution in Eq. (2.14) and the a-priori estimates of ballistic factors and residual variances. After data has been processed for one solar rotation (27-28 days), new ballistic factors and variances are calculated as described above, and we return to the beginning of the solar rotation to reconstruct the density variation models. Usually, only three to four iterations are necessary to meet acceptable convergence criteria [1]. Nazarenko did encounter some problems with convergence, but found the algorithms to be more robust if the global correction factor ξ is applied only in the first iteration. We will follow his recommendations and use this approach hereafter.

2.2.2 Ballistic Factor Estimation with Multiple Standard Satellites

One possible source of error associated with the previous method is that the information obtained from Q_e is associated with a particular altitude, $\overline{h}_e$, which is equal to the perigee altitudes h_{ej} averaged over the spans $j \in N_e$. It is likely that the correction factor ξ is not entirely appropriate for the entire range of altitudes associated with all of the non-standard satellites. However, if a sufficient number of standard satellites with different perigee altitudes can be found, we can construct an altitude-dependent function using the values of Q_e for $e \in I_e$ as measurements. The number of standard satellites will not be large, meaning that the unknown function is assumed to be linear. This assumption ensures that we are not trying to extract too much information from the data, and allows for use of the analytical least square method.

Each value of Q_e is measured in the presence of WGN:

$$Q_e = F(\overline{h}_e) + \varepsilon_e \tag{2.35}$$

The unknown function $F(h)$ is assumed to take the following form:

$$F(h) = a_1 + a_2 \cdot \frac{(h - 200)}{200} \tag{2.36}$$

The coefficients are estimated in a similar manner as the density variation model coefficients in Section 2.1.4, and the equations will not be presented here. We calculate

the RMS error of the residuals and compare to the coefficient a_2. If the absolute value of a_2 exceeds the RMS error, we will assume that $a_2 = 0$ and recalculate a_1. After the coefficients are estimated, the first-stage ballistic factor correction factors can be calculated for each satellite using the following equation:

$$\xi_i \approx 1 + \frac{F(\bar{h}_i) \cdot |N_i|}{\sum_{j \in N_i} \left(1 + \sum_{q=1}^{2} \hat{b}_{qj} f_q(h_{ij}) \right)} \tag{2.37}$$

These correction factors are applied only for the first iteration. The second-stage correction factors described in Eq. (2.32) are calculated in the same manner regardless of number of standard satellites.

2.3 Forecasting of Density Variations

We have outlined approaches for obtaining the linear density variation models for spans for which we have ballistic factor observations. We will now derive the methods used to forecast the variation models for those spans for which we do not have observations. The basic concept is to treat the model coefficients b_{1j} and b_{2j} calculated for each of the N spans as measurements of stochastic processes. The derivation for each of the two coefficients is essentially the same, so we will use the general expression $x(t)$ for both of the processes. We can separate $x(t)$ into deterministic and random components:

$$x(t) = x_d(t) + x_r(t) \tag{2.38}$$

The random component is assumed to be a wide-sense stationary Gaussian random process and has a correlation function defined by

$$K_{x_r}(\tau) = \sigma_{x_r}^2 \cdot \exp(-\alpha|\tau|) \tag{2.39}$$

The variance of $x_r(t)$, given by $\sigma_{x_r}^2$, is in the range of 0.1 – 0.6; Nazarenko [14] outlines the method of calculation in one of his earlier works. The parameter α has been

experimentally determined to be 0.241/day. The corresponding power spectral density is given by

$$S_{x_r x_r}(s) = \frac{2\sigma_{x_r}^2 \alpha}{\alpha^2 - s^2} \tag{2.40}$$

We can graphically conceptualize $x_r(t)$ as the output of a causal[1] shaping filter with unit variance Gaussian white noise as the input:

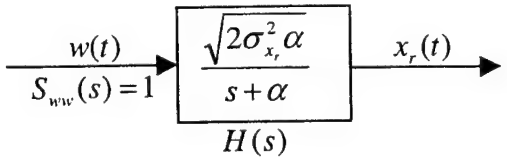


Figure 2.3: Shaping Filter for Random Forecasting Component

This allows us to write the differential equation describing $x_r(t)$:

$$\frac{dx_r(t)}{dt} = -\alpha \cdot x_r(t) + \left(\sqrt{2\sigma_{x_r}^2 \alpha}\right) \cdot w(t) \tag{2.41}$$

To forecast values of $x_r(t)$, we simply solve the differential equation without the white noise (which we have no way of predicting):

$$\hat{x}_r(t) = \exp(-\alpha(t - t_o)) \cdot \hat{x}_r(t_o) \tag{2.42}$$

Therefore, to determine the best estimate of $x_r(t)$ at any future time t, we simply need the initial value at time t_o, $\hat{x}_r(t_o)$. The deterministic component of the signal is assumed to be the sum of a constant parameter and a sinusoid:

$$x_d(t) = \bar{x} + (x_d(t_o) - \bar{x}) \cdot \cos(\lambda(t - t_o)) + \frac{\dot{x}_d(t_o)}{\lambda} \sin(\lambda(t - t_o)) \tag{2.43}$$

---

[1] A linear time-invariant system is *causal* if the output is dependent on the current and/or past values of the input signal. A *causal filter* must have all poles in the left-half plane (LHP) if it is to be stable.

where $\lambda = 2\pi/T$ and $T \approx 27$ days (one solar rotation), $\bar{x}$ is the constant parameter, and $x_d(t_o)$ and $\dot{x}_d(t_o)$ are the values of the function and its derivative at some initial time t_o.

We will be "observing" $x(t)$ at the beginning of each density variation model span, where the beginning time of the j^{th} span is denoted by T_j. We assume that each measurement of $x(T_j)$ is subject to zero-mean white Gaussian noise, denoted by v_j, which has a variance of σ^2 and is statistically independent of $x_r(t)$. Using the traditional designation of $z(T_j) = z_j$ for the measurements, we have

$$z_j = x_d(T_j) + x_r(T_j) + v_j \tag{2.44}$$

Our task, then, is to estimate $\bar{x}$, $x_d(t_o)$, $\dot{x}_d(t_o)$, and $\hat{x}_r(t_o)$, where t_o will be taken as the time of our last measurement, i.e. $t_o = T_N$. Thus, t_o is not actually the "initial" time, but because we are modeling the deterministic component using periodic functions, the temporal location of t_o should not significantly affect results. Because the random component is independent of the deterministic component, we can first estimate $\bar{x}$, $x_d(t_o)$, and $\dot{x}_d(t_o)$ and use the residuals in the second stage to estimate $\hat{x}_r(t_o)$. We can rewrite Eq. (2.44) as

$$z_j = x_d(T_j) + y_j \tag{2.45}$$

where y_j is considered to be the total error of the measurements. If y_j is white Gaussian noise, we can estimate the deterministic parameters using the least-squares (max likelihood) method. However, Eq. (2.39) indicates that the error is correlated at different time moments, meaning that the covariance matrix for the vector of z_j measurements will not be diagonal. Inversion of the covariance matrix for hundreds of measurements (needed in order to calculate the weighting matrix in the least-squares problem) could be very computationally costly. Therefore, we need to make some kind of simplifying assumption. We note that the total interval of measurements processing is one to two months, while the interval of correlation of measurements (controlled by the value of α in the correlation function) is on the order of a few days. This means that we can safely neglect the correlation of different measurements, and because we trust each

measurement equally, can completely eliminate the need to calculate a weighting matrix. The solution to the least-squares problem is found from

$$\begin{bmatrix} \bar{x} \\ x_d(t_o) \\ \dot{x}_d(t_o) \end{bmatrix} = \left(\mathbf{G}^{\mathrm{T}}\mathbf{G}\right)^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{Z} \tag{2.46}$$

where the j^{th} row of the (N x 3) matrix $\mathbf{G}$ is defined by

$$\left|1 - \cos(\lambda(t_j - t_o)) \quad \cos(\lambda(t_j - t_o)) \quad \sin(\lambda(t_j - t_o))\right| \tag{2.47}$$

and $\mathbf{Z}$ is formed from the measurements of z_j, where j = 1,...,N. Once we have the solutions for the deterministic parameters, we calculate the residuals using the following equation:

$$y_j = z_j - \bar{x} - (x_d(t_o) - \bar{x}) \cdot \cos(\lambda(t - t_o)) - \frac{\dot{x}_d(t_o)}{\lambda}\sin(\lambda(t - t_o)) \tag{2.48}$$

We now have N measurements of a stochastic process with additive white Gaussian noise:

$$y_j = x_r(T_j) + v_j \tag{2.49}$$

The optimal solution for this problem is to use a scalar Kalman filter. Eqs. (2.41) and (2.49) function respectively as the state equation and output equation in our state-space model. We use the traditional designations for estimator variance at update time ($p_{j|j}$) and predicted variance ($p_{j+1|j}$), as well as for the current estimate ($\hat{x}_{r,j|j}$) and predicted estimate ($\hat{x}_{r,j+1|j}$). The filter is initialized with $p_{1|0} = \sigma^2$ and $\hat{x}_{r,1|0} = 0$, and the time between the j^{th} and $(j+1)^{th}$ measurement is denoted by τ_j. For any given iteration, we first compute the traditional Kalman gain:

$$g_j = \frac{p_{j|j-1}}{p_{j|j-1} + \sigma^2} \tag{2.50}$$

We then update the last prediction using the new measurement at time j, and propagate the estimator variance:

$$\hat{x}_{r,j|j} = \hat{x}_{r,j|j-1} + g_j(y_j - \hat{x}_{r,j|j-1}) \tag{2.51}$$

$$p_{j|j} = g_j \sigma^2 \tag{2.52}$$

The signal and variance can be propagated using the basic Kalman filter prediction equations:

$$\hat{x}_{r,j+1|j} = \exp(-\alpha\tau_j) \cdot \hat{x}_{r,j|j} \tag{2.53}$$

$$p_{j+1|j} = \exp(-2\alpha\tau_j) \cdot p_{j|j} + (1 - \exp(-2\alpha\tau_j)) \cdot \sigma_{x_r}^2 \tag{2.54}$$

The time span j is incremented, and the equations are iterated until we reach the final measurement at $j = N$. The value of $\hat{x}_{r,N|N}$ is taken to be $\hat{x}_r(t_o)$ in Eq. (2.42), and we can now forecast the random component as necessary. Similar methods have been outlined in the DFY 97 Stage 3 of Nazarenko's report [1].

Chapter 3 Tools and Software

A key requirement of atmospheric density correction in near-real time is an efficient and high-speed method of organizing and processing large amounts of data. In addition, the estimation of ballistic factor observations must occur in an automated fashion, as hundreds of objects are used as calibration satellites. Two software tools proved to be absolutely essential for the work in this thesis: the GTDS computer application, and scripting files written in the Perl 5.0 programming language. Each is described in the sections to follow.

3.1 The Research and Development Version of the Goddard Trajectory Determination System (R&D GTDS)

In order to make the best use of our observations, and also to be able to separate the effects of atmospheric drag from other perturbations, we must use a propagation model that is as accurate as possible. One of the most accurate special perturbation models available is the Research & Development version of the Goddard Trajectory Determination System (R&D GTDS) as currently implemented at the Charles Stark Draper Laboratory.

R&D GTDS is a multi-purpose computer application originally developed for NASA/Goddard Space Flight Center in the 1970s. The R&D version has been extensively modified at Draper Laboratory to include Precision Mean Element-based orbit generation and determination. A sequential Kalman filter (SKF) and extended sequential Kalman filter (ESKF) were added to estimate mean elements. New observation models and coordinate systems were added; the NORAD GP Theory and the Naval Space Command PPT2 Theory were included as perturbation models. R&D GTDS is divided into nine functional components, but only three will be used in an atmosphere correction service. They are:

Ephemeris Generation Program (EPHEM) – This utility is used to propagate satellite states from epoch over a given period of time. Nineteen analytic and

numerical propagation techniques can be called by EPHEM. The technique used for atmospheric correction will be high-precision numerical integration, which usually goes by the name of Cowell's method when applied to orbit propagation problems. The particular implementation of Cowell's method respectively employs multi-step Störmer-Cowell and Adams methods for solution of the class II and class I differential equations. EPHEM will generate state histories in binary or ASCII format upon request.

Differential Corrections Program (DC) – The DC program uses simulated or real observations to estimate the values of desired solve-for parameters such as orbital elements, dynamic coefficients, and station locations and biases. A method is chosen to propagate the a-priori state, and residuals are computed from the difference between predicted and input observations. The residuals are used to solve the linearized system for the change in state, and the process is repeated until the state parameters converge to a specified tolerance. The program can also generate statistics such as correlation coefficients, mean values, and covariance matrices.

Data Simulation Program (DATASIM) – The DATASIM program uses a binary file generated by EPHEM containing satellite states over the desired time span. This file is used to create simulated observations of a specified type and at a desired frequency. Observations can include equipment and environment-dependent biases and noise. The observations are computed using any desired ground station location or satellite location if satellite-to-satellite tracking (SST) is used. These observations can be used as input to the DC program or to determine visibility or tracking schedules.

Fischer [37] details the history and capability of the many currently existing versions of R&D GTDS, and his work will not be repeated here. It will be instructional, however, to describe the various fixes and modifications made to the version of GTDS used in this thesis. When work on atmospheric corrections at Draper first began in 1999,

several requirements for an orbit propagator were identified. The propagator had to be very accurate, computationally efficient, and include numerous perturbation models. It also had to include an accurate drag model able to effectively capture long-term trends in atmospheric density, such as JR-71 or MSISE-90. Finally, it was desired to implement the software on a UNIX-based SGI workstation for greater computational speed and stability. These requirements left us with two possible starting points: VAX-GTDS and NT-GTDS PR6. The former was implemented on a Digital Equipment Corporation (DEC, now owned by Compaq) VAXStation 4000 with an OpenVMS V6.2 operating system. The latter was ported from an SGI system to the 486 personal computer environment in 1994, and then to the 32-bit Windows NT environment in 1996. Each version contained functionality lacking in the other, but NT-GTDS PR-6 included both the JR-71 and MSISE-90 atmospheric models, and it was thought that PR-6 would be easier to port back to the UNIX environment. NT-GTDS PR-6 was therefore selected as the platform for further development. Before work on atmospheric correction could begin, however, NT-GTDS had to be validated as an error-free platform for high-accuracy orbit determination and prediction.

3.1.1 Validation of NT-GTDS

Metzinger [38] developed a series of test cases in 1993 designed to fully validate each component and function of GTDS, regardless of platform. These test cases had been executed for the PR-2 version of GTDS, which was an earlier version also implemented on the PC; however, the test cases had not been executed in a systematic fashion for any version since. It was hoped that PR-6 would prove to be fully functional as it contained a number of improvements dealing with sequential filtering and complex drag modeling not available in other versions. Unfortunately, when a differential correction (DC) run was performed using actual NORAD observations of SV10299 in Test#3, significant errors arose. These errors did not occur when the same test was run using an earlier version of NT-GTDS designated by PR-5. Due to the complex nature of GTDS and the fact that at a minimum PR-5 contained the MSISE-90 atmospheric model, it was decided to abandon PR-6 and attempt to validate PR-5 as the development platform instead of attempting to debug PR-6. The test cases were repeated on the PC,

and all 21 tests were shown to match Metzinger's results to acceptable levels of accuracy (mm level or better).

One problem identified with the development of GTDS on the PC was that some changes were made to the source code which went undocumented; this made it very difficult to find and remove errors discovered after the fact. Based on our experiences with NT-GTDS, we decided to set a requirement for UNIX-GTDS that it be included in a version control system, in which changes must be documented and original versions of code can always be retrieved. This requirement should greatly simplify any future debugging or modifications should they become necessary.

3.1.2 Validation of UNIX-GTDS

A total of 1320 FORTRAN source files were copied from the PC to DC1, an eight-processor Silicon Graphics Inc. Origin 2000 machine running IRIX 6.5.3. The source code was immediately imported into the Concurrent Versions System (CVS) 1.10.5 version control system [54] to track changes and to preserve the original importation. A makefile was constructed and, after a few small modifications, the code was compiled and linked into an executable. The necessary compilation steps and platform-dependent options are described in Appendix B.

The next major task was to construct the many data files that are required for execution of GTDS. Many of these data files are stored in binary format, and therefore cannot be directly transferred from system to system, but must instead be recreated in each location. A library of data file generation routines was constructed, and the ten or so data files were generated and linked to appropriate GTDS input files. Particular attention was paid to GTDS$075, which is the input file required for the JR-71 atmospheric model. This data file is built from inputs from the National Geophysical Data Center (NGDC) [39] in the form of daily values of $F_{10.7}$ and Ap. Where these inputs are not available, such as for runs requiring propagation of orbits into future times, the data file can be built from predicted indices provided by Dr. Kenneth Schatten at the National Science Foundation [40]. Fischer describes the building of the JR-71 and JR-71/MSIS (GTDS$076) files in detail in his thesis, but a few modifications and fixes should be noted here. First, an implicit conversion of real data to integer data on the PC was

leading to slightly inaccurate values of Ap. A more significant error was found in the interpolation of real data into the Schatten data. An error in integer math was causing the GTDS file to use all real data until the beginning of the Schatten file, at which point the values jumped in a discontinuous fashion to the predicted values. This bug was fixed in such a manner as to allow smooth progression from real data to Schatten data over the 81-day interpolation region. The building of the GTDS data files is described in more detail in Appendix B.

With the code compiled and the data files built, the 21 Metzinger test cases were executed. All cases performed as expected, with the exception of some output file options, and the UNIX port of PR-5 was validated as the version for future development. This does not mean that the code is entirely error free in the SGI environment, however; a list has been maintained of desired additions and fixes, and is presented in Appendix B for future reference.

3.1.3 Incorporation of Density Correction into UNIX-GTDS

To allow for demonstration of the effectiveness of the density correction process, GTDS was modified to read a text file of variation model coefficients and calculate corrections upon request. There were a number of tasks involved in this modification, including adaptation of existing code and incorporation of a few new routines. A GTDS Control Card was also added to allow a user to turn on or off atmospheric correction without having to recompile any code. The code modifications, additions, and new variables can be referenced in Table 3.1 and Table 3.2 and the new control card in Table 3.3 below.

3.1.3.1 Modifications to Existing Code

The first change was to the SETDAF.FOR subroutine, which opens all data files necessary for the particular type of run requested. Fortran Reference Number (FRN) 106, which is assigned to the file GTDS$106, was associated with the JR-71 atmospheric correction file. It should be noted that each correction file is specific to a particular model, so all modifications have been made in such a way as to allow for other correction files (perhaps for the MSISE-90 model or GOST) to be added later. The FRN of the JR-

71 correction file was assigned to a common block in FILESBD.FOR, and the SHUTDAF.FOR routine was modified to close the new data file upon job completion.

The next necessary step was to modify SETORB.FOR and SETOG1.FOR to allow for reading of the new Control Card, which was identified with the label "ATMCAL". The ATMCAL card, which is described in detail below, allows a user to turn on or off atmospheric correction of a specified density model for EPHEM or DC runs as desired.

The atmospheric density is calculated in a number of different routines in the JR-71 model, depending on the requested altitude; therefore, the optional call to the density correction routine had to be placed in each location. If the altitude is between 90 and 100 km, the BARODE.FOR routine is used; for 100-125 km, the DIFFDE.FOR routine; and for greater than 125 km, the density is calculated in HIALT.FOR. An optional call to CALCCALJAC.FOR was placed in each location. Also, the density correction common block (described below) had to be initialized at the start of each run, so a call to INITCALJAC.FOR was added to JACROB.FOR, which is the controlling routine for calculation of density in JR-71.

3.1.3.2 New GTDS Subroutines

Two new routines, a block data file, and a ".cmn" initialization file were added to the GTDS code. The ".cmn" file defines the /ATMCALJAC/ common block with a number of new variables listed in Table 3.2 below. The new routines are INITCALJAC, which reads the correction coefficients into the common block; CALCCALJAC, which calculates the density correction based on input request time and altitude; and ATMCALJACBD, which initializes the /ATMCALJAC/ common block.

Table 3.1: GTDS Code Modifications/Additions For JR-71 Atmospheric Correction

Routine	*Modification/Addition*
ATMCALJACBD	Initializes JR-71 density correction variables in /ATMCALJAC/ common block
	Initial addition to GTDS
BARODE	Calculates JR-71 density for 90-100 km
	Added call to CALCCALJAC

CALCCALJAC	*Calculates density corrections for the JR-71 model*
	Initial addition to GTDS
DIFFDE	*Calculates JR-71 density for 100-125 km*
	Added call to CALCCALJAC
FILESBD	*Defines FRNs for standard GTDS input/output files*
	Added FRN 106 for Jacchia atmospheric correction file
	Set IFILE(106) in /FILES/ common block equal to NCALJAC
	Set NCALJAC = 106
HIALT	*Calculates JR-71 density for above 125 km*
	Added call to CALCCALJAC
INITCALJAC	*Reads Jacchia density correction file into common block*
	Initial addition to GTDS
JACROB	*Driver routine to calculate JR-71 density*
	Added optional call to INITCALJAC.FOR
SETDAF	*Opens all necessary GTDS files*
	Opened GTDS$106 – Jacchia correction file
	Added 'READONLY' to data file opens for multi-user access
SETOG1	*Interprets orbit gen. cards that come after DRAG in SETORB*
	Added ATMCAL card
	Set ATMCAL switches if read ATMOSDEN card
SETORB	*Interprets orbit generator optional keyword cards*
	Added code to interpret ATMCAL card
SHUTDAF	*Closes all necessary GTDS files*
	Modified to close all FRNs from 1-106

Table 3.2: New Variables in the ATMCALJAC Common Block

Variable	*Description*	*Type*	*I/O*
DATEBEGJAC	Beginning date of JR-71 correction file	Real*8	O
DATEENDJAC	End date of JR-71 correction file	Real*8	O
SPANEPCHJAC	Array of span length for each span j	Real*8	O
CALB1JAC	Array of b_1 correction coefficients	Real*8	O
CALB2JAC	Array of b_2 correction coefficients	Real*8	O
CALSWITJAC	On/off switch for JR-71 correction	Logical	I
CALINITJAC	Specifies whether to initialize common block	Logical	I/O

3.1.3.3 The ATMCAL GTDS Control Card

The following table describes the formatting of the new ATMCAL GTDS control card. The card was designed such that corrections to other atmospheric models, such as

MSISE-90 or Harris-Priester, may be specified at such point when the functionality is added to the code. Note that only the JR-71 option is currently supported.

Table 3.3: ATMCAL Control Card Description

ATMCAL (OGOPT)

- Card format: (A8, 3I3, 3G21.14)
- Applicable programs: DC, EPHEM, FILTER
- Detailed format:

Columns	Format	Description
1-8	A8	ATMCAL – Input card to atmospheric corrections
9-11	I3	Turn on/off atmospheric correction =0 Off (default) =1 On
12-14	I3	Number of atmospheric model to apply corrections to*: =1 Jacchia-Roberts '71 =2 Harris-Priester =3 Jacchia-64 =4 Jacchia-70 =5 MSIS-77 =6-8 Reserved for RADARSAT =9 MSISE-90 =10 Reserved for GOST
15-17	I3	Unused
18-38	G21.14	Unused
39-59	G21.14	Unused
60-80	G21.14	Unused

* JR-71 is currently the only model that corrections may be applied to in GTDS, as of May 2000.

3.1.4 Other Code Fixes and Modifications

A number of other bug fixes and modifications were necessary to make UNIX-GTDS an effective platform for testing of atmospheric density correction. The new routine ASCII_ORB1_DATA was ported from VAX-GTDS and added to UNIX-GTDS to allow for output of .ASCII files in parallel with the binary .ORB1 files. Also, two major bugs were identified and fixed:

1. **No-observations bug**: GTDS crashed if a specified station in a DATASIM run did not have any observations of the target satellite.
2. **Year-rollover bug**: a DC run would not execute properly if observations in the input file crossed a year boundary.

The following table outlines the subroutines that were modified or added to UNIX-GTDS.

Table 3.4: GTDS Code Modifications/Additions For JR-71 Atmospheric Correction

Routine	*Modification/Addition*
ASCII_ORB1_DATA	*Writes .ASCII files in parallel with .ORB1 files*
	Initial addition to GTDS
ELEME	*Converts Cartesian, Keplerian, or spherical elements to one of the other two systems*
	Removed debug print
FILESBD	*Defines FRNs for standard GTDS input/output files*
	Added FRNS 101-105 for ASCII state histories; these files correspond with the five .ORB1 files, with 101 ←→ 24 (primary) and 102 ←→ 81 (secondary).
OBSWF	*Writes observation working file for DATASIM run*
	Fixed year-rollover bug by ensuring EDAY refers to correct year
ORB1	*Writes the .ORB1 binary output files*
	Added call to ASCII_ORB1_DATA
SETDAF	*Opens all necessary GTDS files*
	Opened GTDS$101-105 (.ASCII files associated with primary and secondary .ORB1 files)
	Added 'READONLY' to data file opens for multi-user access
STARPT	*Generates printer summary report of passes in DATASIM run based on information in DSP summary file*
	Added test to ensure num. of records in DSP file > 0

3.2 Atmospheric Correction Driver Programs

One of the most challenging aspects of the methodology presented in this thesis is the amount of computation involved. If accurate atmospheric correction is to be applied for any significant length of time, literally thousands of differential correction jobs are required. It very quickly became clear that an efficient methodology for automating such tasks as truth file generation, data simulation, differential correction, and calculation of density variations was required. It was decided to automate these tasks using script files written in the Perl 5.0 programming language. Perl is an obvious choice due to its ease of use, readability, and speed, as well as for its ability to work with text files. For further documentation, please refer to the Perl programming guide, otherwise known as the "Camel Book" [41]. A brief description of each of the main script files follows, along with some of the design decisions that went into their construction. The first two scripts are necessary only if the atmospheric correction run will be using simulated observations. Chapter Four contains a more detailed description of each data file and the overall data flow for each test case, whereas this section focuses mainly on how to execute the scripts in a general sense.

3.2.1 Generation of Osculating Truth Files: The TLE2osc Program

The first task for a simulated run of atmospheric correction is to generate the "truth" files for all objects. The input file takes the form of a list of satellites in NORAD's two-line element (TLE) format [42]. The "true" ballistic factor is converted from the NORAD drag parameter (BSTAR) given on the two-line element set. The list of TLEs should be as close to the desired start epoch as possible, and should only contain satellites with perigees in the proper altitude range (200-600 km). The user must specify the name of the input and output files, start and end epochs, and how many of the input objects are to be "standard" satellites. Because Perl scripts can be run without manual compilation, these options can be changed directly in TLE2osc.pl. The program creates a GTDS card deck, runs GTDS, and produces an .ORB1, .ASCII, and .ORBIT "truth" file for each input object in the list of TLEs. The program also writes the initinfo.txt file, which contains a list of the NORAD Space Surveillance Center (NSSC) catalog numbers

for each satellite to be used for atmospheric calibration. Other information needed for the subsequent phases of the simulation is also contained in this file. The exact structure of initinfo.txt is described in more detail in Chapter 4.

3.2.2 Generation of Simulated Observations: The genobs Program

This program relies on the GTDS DATASIM program to simulate all observations. DATASIM uses the .ORBIT file and the initinfo.txt file created by TLE2osc.pl to generate range, azimuth, and elevation observations from user-specified ground stations. The user must also provide the names of input and output files and begin and end epochs. The program determines which objects are standard from initinfo.txt, and adds Gaussian noise to the observations accordingly. genobs.pl outputs an observation file for each object in OBSCARD (GTDS$015) file format, where each observation is printed with epoch on a separate line in a sequential .ASCII file.

3.2.3 Estimation of Short-Arc Ballistic Factors: The estbfs Program

This script is one of the most important and complex components of the density correction process, since it must be executed regardless of whether the observations are real or simulated. The main purpose of the program is to automatically cycle through the overall time interval in sequenced spans of three to four days and execute short-arc fits of observation data for each object Because thousands of GTDS DC runs are required and each run can take anywhere from 5-30 seconds, it was decided to give estbfs.pl the capability of spawning multiple processes to break up the job into smaller pieces. This is only possible on a platform such as the SGI DC1 machine that has multiple processors and multitasking capabilities.

The program first reads initinfo.txt to determine which objects are to be used for atmospheric correction. It then sets up a card deck for the first differential correction run starting from the specified start epoch. The length of the fit span is also an input option. A key issue is where to obtain the a-priori state vector for the DC program. If the observations are simulated, the a-priori state vector for the first run is taken from the truth file and from converged DC runs thereafter. If the observations are real, the first a-priori

estimate must be derived from some outside source, which will usually be an up-to-date TLE set; subsequent a-priori state vectors will again be taken from converged DC runs.

Another important consideration is the a-priori ballistic factor. For simulated objects, if the object is standard, estbfs.pl provides the true ballistic factor given in initinfo.txt; otherwise, the true ballistic factor is randomly distorted by up to a factor of two. For real objects, the only change for standard or non-standard objects is in setting the a-priori standard deviations in the differential corrections process.

The DC program is executed and the output tested for convergence. If the run converges, the resulting estimated ballistic factor (designated by k_i in earlier chapters) is stored in ballfcts.txt, and the propagated Cartesian state is used as an a-priori guess for the next DC run. The fit span is shifted by a default number of hours (usually three), and the process continues until the end of the overall time span as specified by the user. One more function of estbfs.pl is to generate an observation schedule for each object; if there are no new observations for a particular fit span, by default the DC program is not executed.

3.2.4 Calculation of Density Variations: The calcvars Program

This script performs the actual calculations of density variations, estimation of "true" ballistic factors, and forecasting of variation coefficients as necessary. The majority of the code is written in Perl, but the matrix manipulations and filtering are written in Matlab code and called as a Matlab script file from calcvars.pl [57]. As has been the case for the other script files, the initinfo.txt file is used to provide input information such as true ballistic factors and which satellites are considered to be standard. The calcvars.pl script also requires ballfcts.txt from estbfs.pl, and input options such as desired begin and end epoch and file locations. The user must also specify if calculation of "true" ballistic factors is desired, in which case iteration between calcvars.pl and estbfs.pl will occur. The program will output a file with a user-supplied filename that will contain density variation coefficients in a format recognizable by UNIX-GTDS.

3.3 Other Data Processors and Program Utilities

A few other miscellaneous programs were useful in various stages of this research and are documented below.

3.3.1 TLE Processing Utilities

A user often wishes to extract a sub-set of objects from a long text file in TLE format. A convenient utility written by Willie Koorts and available on his home page [43] was used for such a purpose. The extract.exe program takes an input file of NSSC numbers or names of satellites and the original TLE text file, and outputs the corresponding subset of TLEs.

Another useful capability is to be able to screen objects in a TLE file for desired characteristics such as perigee height or eccentricity. Mike McCants has written a program entitled xlate.exe, also available on his home page [44], that takes a TLE file and outputs a list of objects in more readable format. The user can specify ranges of period, mean motion, apogee, perigee, or other orbital characteristics. The above two utilities are stored on the PC under the G:\GRANHOLM\THESIS\TLES directory on an archived hard drive belonging to Dr. Paul Cefola at Draper Laboratory.

Throughout the atmospheric correction process, it is often necessary to make conversions from calendar date to Julian date and vice versa. Two routines, cal2jul and jul2cal, were written in Perl and included in each Perl script as the "Dates" module. The conversion routines have been validated to better than 0.0001 seconds accuracy.

3.3.2 Observation Processing Utilities

NORAD provides its observations to users in B3 format, which must be converted to OBSCARD format if to be used by GTDS. Lt. Col. David Vallado wrote a utility to perform this conversion which is called convobs.exe; readers interested in obtaining a copy of the program should contact Lt. Col. Vallado directly at (719-554-3638) or via e-mail at valladod@usspace.cas.spacecom.af.mil.

[This Page Intentionally Left Blank]

Chapter 4 Data Flow and Task Description

This chapter will describe the inputs and outputs of each part of the density correction process, and will outline the different test stages and types of test cases. The first section deals with the simulated observations case and is followed by a section on real observations. The third section goes into the test cases to be executed for the following test stages: concept validation, correction of inaccurate density models, and forecasting of model coefficients.

4.1 Detailed Data Flow for Simulated Observations

Presented below is a data flow diagram showing the interface between the utilities used in the atmospheric correction process for simulated observations. Each step in the process is explained in more detail in the sections that follow.

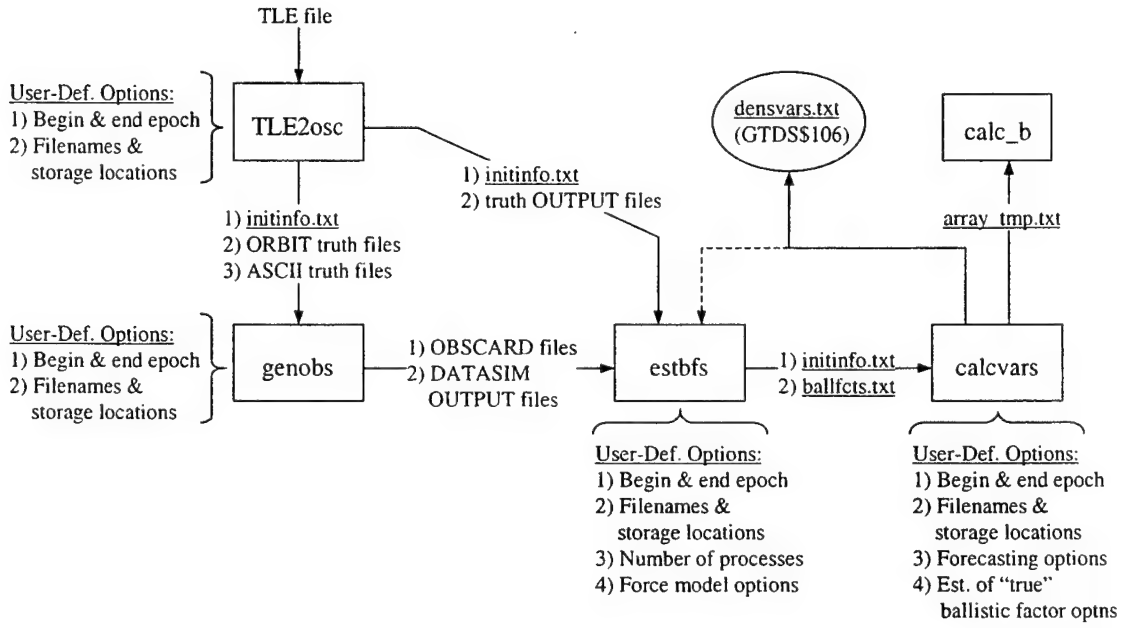


Figure 4.1: Program Utility Data Flow For Simulated Data

4.1.1 Generation of Osculating Orbits

For the simulation to be as realistic as possible, the distribution of calibration satellites should be approximately equivalent to the actual distribution of LEO objects currently tracked by NORAD. The best way to obtain a realistic distribution is via the processing of a current listing of the space catalog in TLE format, which is available from a number of locations on the Internet. We were able to obtain a fairly comprehensive list from Allen Thompson's compilation on the Jet Propulsion Laboratory anonymous FTP site [49].

The list of TLEs can be parsed for perigee height using both the xlate.exe and extract.exe programs described in Chapter 3, with a constraint of 200 km < h_p < 600 km. Another constraint should be placed on apogee height so as to eliminate satellites that spend a majority of their orbits above 600 km; suggested allowable values fall between 200 and 800 km. This constraint theoretically allows for a maximum eccentricity of 0.044, but in practice the eccentricity for these type of objects rarely exceeds 0.03.

The TLEs are originally generated by NORAD using their SGP4 general perturbation theory [55], and must be converted to osculating elements to allow for high-precision propagation. Before this conversion can take place, however, the elements must be formatted for input into a GTDS card deck. Implicit in this formatting is a time conversion from NORAD day of year to Julian date. NORAD labels each two-line element with a two-digit year and a fractional day of year, where 99001.000 would be 0:00 UT January 1st, 1999. This time format implies potential confusion for dates after 1999, but NORAD has specified that all year fields greater than 56 refer to years before 2000, and all year fields less than or equal to 56 refer to years of 2000 or later. This problem will have to be resolved on a more permanent basis in 2056, but hopefully improvements will be made in space catalog maintainance over the next fifty years that will make NORAD SGP4 theory obsolete.

The other initial conversion that must take place deals with the SGP4 drag parameter, BSTAR. BSTAR is given in units of ER^{-1} (inverse Earth radii), and is converted to the conventional definition of ballistic factor given in Chapter 2 using the following formula taken from Vallado[50]:

$$k = 6.3708105 \cdot BSTAR \tag{4.1}$$

Here, k is in units of m^2/kg. Note that a different "ballistic factor" is defined by Vallado and others without the ½ term, so the conversion factor has been scaled accordingly.

We need a few more pieces of information before being able to make the conversion to osculating elements. The EPHEM program requires that we separate the ballistic factor into its components of cross-sectional area, mass, and C_D. We repeat the definition of k from Chapter 1:

$$k = \left(\frac{C_D A_x}{2m} \right) \tag{4.2}$$

The actual drag force is computed using only k, so as long as the product of area-to-mass ratio and $C_D/2$ remains the same, we may assign the individual values somewhat arbitrarily. However, the area-to-mass ratio is also used in computation of solar radiation pressure, so it is desired to use a somewhat realistic estimation. If we assume some nominal value for C_D and use the radar cross-section (RCS) as an estimate of A_x, we can solve for the mass using Eq. (4.2). A good average value of C_D for LEO satellites is 2.2, as shown by numerous studies [4,51].

The general perturbation theories employed by NORAD have been shown to be relatively inaccurate, and even more so for low-altitude objects [52]. Therefore, it is desired to propagate using NORAD theories for as short a time as possible, and then use high-accuracy special perturbation techniques for the remaining long-arc truth file generation. GTDS performs the coordinate system conversion from NORAD Historical Data System format for the TLEs (type 18 on the ELEMENT1 card) to mean Earth equator and equinox of 1950. The latter is the coordinate system used for the remainder of the data processing.

These calculations and conversions are performed by the TLE2osc.pl program introduced in Chapter 3. The program automatically does TLE to osculating conversions for all TLEs given in the input file. The primary output of a TLE2osc.pl run is a file called initinfo.txt, which contains the NORAD Space Surveillance Center (NSSC) catalog number, international COSPAR/WWAS (COSPAR World Warning Agency for

Satellites) designation, true ballistic factor k in m^2/kg, radar cross-section in m^2, true variance σ_i^2, standard or non-standard status, and a designation for observation type (simulated = 29, real = 15). The file is structured as follows:

Table 4.1: Format of initinfo.txt

NSSC#	Int'l Des	k_i	RCS	σ_i^2	Stnd/Non	Obs. Type
00063	60016A	1.97068E-03	0.523	1.0000E-06	S	29
00179	61015BD	9.03126E-02	1.155	2.0000E-06	N	29
:	:	:	:	:	:	:

The format of this file is somewhat similar to a data file used by Nazarenko and labeled as Table 2 in DFY 97 Stage 1 of his report [1], but more information such as international designation and RCS has been added to accommodate GTDS requirements. The a-priori variance of each object is arbitrarily assigned a different default value for standard and non-standard objects, where the non-standard variance is twice the standard variance. The actual values of variances do not matter in themselves, because they are used only for relative weighting in the least-squares estimation of density variation coefficients. It is only after the first iteration in the estimation of "true" ballistic factors that values of σ_i^2 will change for each satellite.

Additional outputs of EPHEM used by the other density correction utilities are: ORBIT files (GTDS$020), which are direct-access binary files containing all necessary information for a DATASIM run; ASCII files (GTDS$101-105), which can be used as "truth" state histories in later test cases; and the OUTPUT files (GTDS$006), which are used by estbfs.pl to come up with a-priori state vectors in the DC runs.

4.1.2 Generation of Simulated Observations

For all simulation runs in this thesis, observations were assumed to come from four ground station locations: Eglin AFB, FL; Kaena Point, HI; Fylingdales, England; and Grand Forks, ND with the PAR (Safeguard) system. These locations were chosen to approximate real-world tracking geometry and observation rates. The observation scheduling is executed such that the average number of observations per object per day

approximately matches the number of real observations obtained by NORAD for low-altitude space objects. Because Fylingdales, Grand Forks, and Eglin are phased-array radars, they are given twice the observation rate of Kaena Point. Another requirement is that the flow of observations should be evenly distributed throughout the day, so that there will be enough $\hat{k}$ estimations to construct each three-hour density variation model.

Range, azimuth, and elevation observations can be simulated with or without noise as desired. Noise characteristics and station locations are taken from the Station Location and Accuracy Database (SLAD) compiled by J. Fischer for his thesis research [37]. This information is not reproduced here, but if more information is desired, the reader can contact Dr. Ronald Proulx and Dr. Paul Cefola at the Draper Laboratory.

GTDS incorporates station locations and noise information using Station Cards 1 and 0, respectively. It is also possible to set default noise characteristics for a particular type of observation across all stations using the OBSDEV card, but the reader should note that Station Card 0 takes priority.

The genobs.pl program reads initinfo.txt, sets up GTDS card decks for all given objects, and executes DATASIM runs with desired options. Initially, the observations were output in the form of the GTDS$029 binary file for later use by the DC program. However, a bug was discovered dealing with the OBSINPUT card: the beginning epoch on OBSINPUT must match the beginning epoch of GTDS$029, or the code will not execute properly. This means that in order to specify the correct observation input span, the user must include an ACCREJ card with the start and end times of the fit span. However, the ACCREJ card cannot be placed in the DCOPT subdeck as the code will again crash; instead, ACCREJ must fall under the DMOPT subdeck. This is very inconvenient if the solve-for epoch is in the middle of the overall time interval. Because GTDS first builds an observation working file containing all observations up until the solve-for epoch, execution time may increase by an order of magnitude.

The solution to this problem is to output all observations in GTDS$006, the default output file, and then program the script file to automatically build an observation file in OBSCARD (GTDS$015) format after the DATASIM run finishes. The OBSCARD observation files are in more readable form (.ASCII), and fortunately do not exhibit any of the undesirable features associated with GTDS$029.

4.1.3 Differential Correction and Generation of $\hat{k}$ Measurements

The only input data files required for the execution of the DC runs are the initinfo.txt file and the OBSCARD file for each object. However, the estbfs.pl program also needs OUTPUT files from the EPHEM runs to obtain an a-priori state estimate for the first DC run, and from the DATASIM runs to generate observation schedules for each object. After the first DC run, a-priori guesses for the state vectors are taken from the OUTPUT files of converged DC runs. For the a-priori ballistic factors, if the object is standard, the a-priori is equal to the truth. If the object is non-standard, the true ballistic factor is distorted by a random factor with probability ½ of falling between 0.5 and 1 and probability ½ of falling between 1 and 2. These distorted values of "true" ballistic factors are stored in a new file entitled initinfo_est.txt, which will be used for estimation of "true" ballistic factors in calcvars.pl.

Some objects will not have enough observations to allow the DC program to converge on a viable solution. Therefore, an option has been added to estbfs.pl to move to the next object if DC fails to converge in a specified number of days. The physical model options can be altered from truth options as desired.

GTDS cannot directly solve for k in a differential correction run, but instead solves for the relative variation of the coefficient of drag (C_D) in the form of the variable ρ_1:

$$\hat{C}_D = C_D(1+\rho_1) \tag{4.3}$$

Therefore, a non-zero value of ρ_1 will adjust the default value of C_D up or down as necessary. This estimated value of the drag coefficient is then substituted back into Eq. (4.2) to give a measurement of $\hat{k}$ at the appropriate perigee height and attribution time.

The output of estbfs.pl is stored in a file named ballfcts.txt, which includes the NSSC catalog number, attribution time in Julian date, estimated ballistic factor, and current perigee height in km:

Table 4.2: Format of ballfcts.txt

NSSC#	t_i	$\hat{k}_i$	h_i
00063	2451529.00	2.10366E-03	5.45516E+02
00063	2451529.75	2.12907E-03	5.39491E+02
:	:	:	:

4.1.4 Calculation of Density Variations

The calcvars.pl program initially reads both the initinfo.txt (or initinfo_est.txt if "true" ballistic factors are to be estimated) and the ballfcts.txt files. The ballistic factor estimations are sorted into three-four hour time spans (the default length of each span, τ_{min}, can be defined by the user), and each span is tested to ensure that it contains at least 35 estimations. If necessary, the span length τ_j is extended until the number of estimations exceeds the minimum value. The program then calculates and writes the $\mathbf{F}_j$ and $\mathbf{P}_j$ matrices and the $\mathbf{a}_j$ vector defined in Chapter 2 to a temporary text file, and calls the Matlab script calc_b.m. The Matlab script reads the temporary text file, and performs a 3-σ test on the values in the $\mathbf{a}_j$ vector such that any measurements greater than three standard deviations from the mean are rejected. Finally, Matlab performs the necessary matrix multiplications and inversions to calculate the density variation coefficients b_{1j} and b_{2j}, also defined in Chapter 2. The density variation coefficients are written in a user-defined text file that can be read as GTDS file 106:

Table 4.3: Format of Density Variation Coefficients File

$t_{start,j}$	b_{1j}	b_{2j}
2451529.000	2.40157E-02	1.79451E-02
2451529.125	2.79308E-02	2.49923E-02
:	:	:

If iterative estimation of "true" ballistic factors is requested, calcvars.pl calculates the new estimates of ballistic factors and variances and stores them in initinfo_est.txt for

the next run of estbfs.pl. If forecasting is requested, the program uses the density variation coefficients to predict values up to a specified end epoch.

4.2 Data Flow for Real Observations

The second stage of the density correction process does not have to be executed if real observations are available. The process must be initialized using TLE2osc.pl, which is followed by execution (and iteration) of estbfs.pl and calcvars.pl.

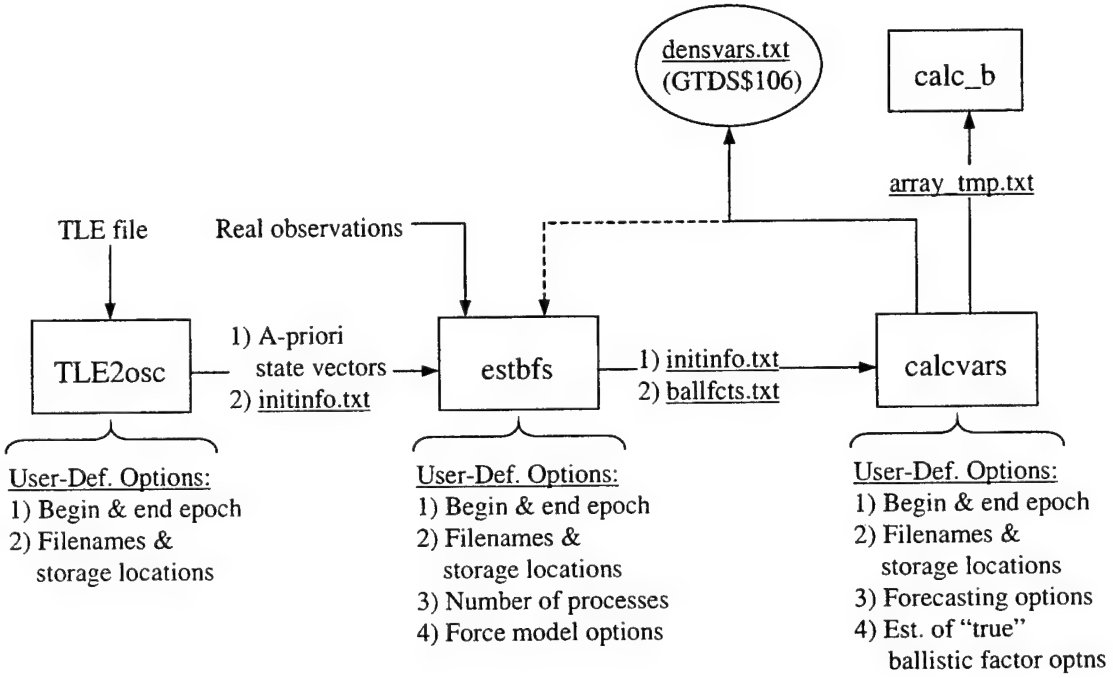


Figure 4.2: Program Utility Data Flow For Real Data

4.2.1 Differential Correction and Generation of $\hat{k}$ Measurements

While it is not necessary to generate truth files when using real data, it will still be necessary to run an abridged version of TLE2osc.pl so as to generate the initinfo.txt file. Standard and non-standard satellites should be assigned based on actual spacecraft characteristics. For those satellites where a "true" ballistic factor is unknown, an up-to-date TLE can be used as a rough estimate. Current TLEs for the selected objects will also be used for the first a-priori guesses of spacecraft states in the DC runs. Observations

should be supplied in OBSCARD-compatible format. After initialization, the estbfs.pl program is run in the same manner as for simulated observations.

4.2.2 Calculation of Density Variations

The calcvars.pl program does not differ in execution for real or simulated jobs. The default will be to estimate "true" ballistic factors, as there will undoubtedly be a large number of objects for which we do not have accurate area, mass, or coefficient of drag.

4.3 Test Cases

4.3.1 End-to-End Software Validation

The purpose of the first test is to validate the flow of data from one piece of the simulation to the next. The observations will be simulated with no noise, and the DC runs will be given the truth density model and truth ballistic factors. The resulting density variations are expected to essentially equal zero over the entire time period.

The test will be run over the first fifteen days of a two-month interval beginning on December 15th, 1999. This period of time was chosen to match the begin and end times of the real data that U.S. Space Command has made available to Draper Laboratory for continuing research in atmospheric correction. Atmospheric conditions over this interval are average: the daily values of Ap exhibit some instability but do not exceed 40, while the $F_{10.7}$ fluctuates around 155 $(W/m^2)/Hz$. Ap values for the time period in question are presented in Figure 4.3 below. The Schatten predict values are also included on this plot for reference in other tests.

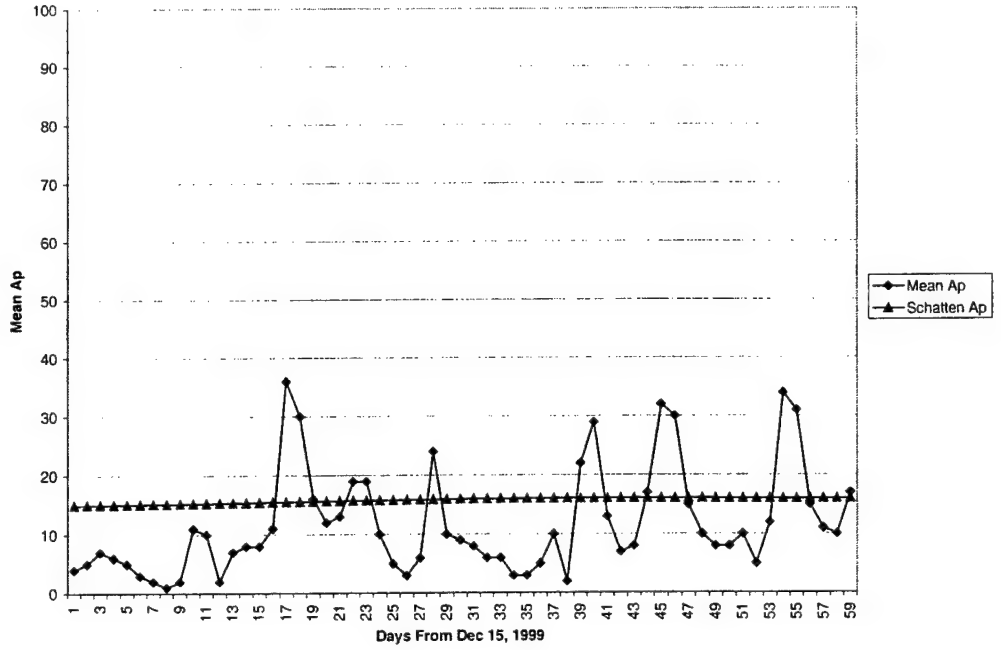


Figure 4.3: Average Planetary Amplitude (Ap), Dec 15, 1999 - Feb 11, 2000

To give some sense of how these geomagnetic conditions compare with other time intervals, the figure below illustrates a particularly perturbed six-month period in 1992:

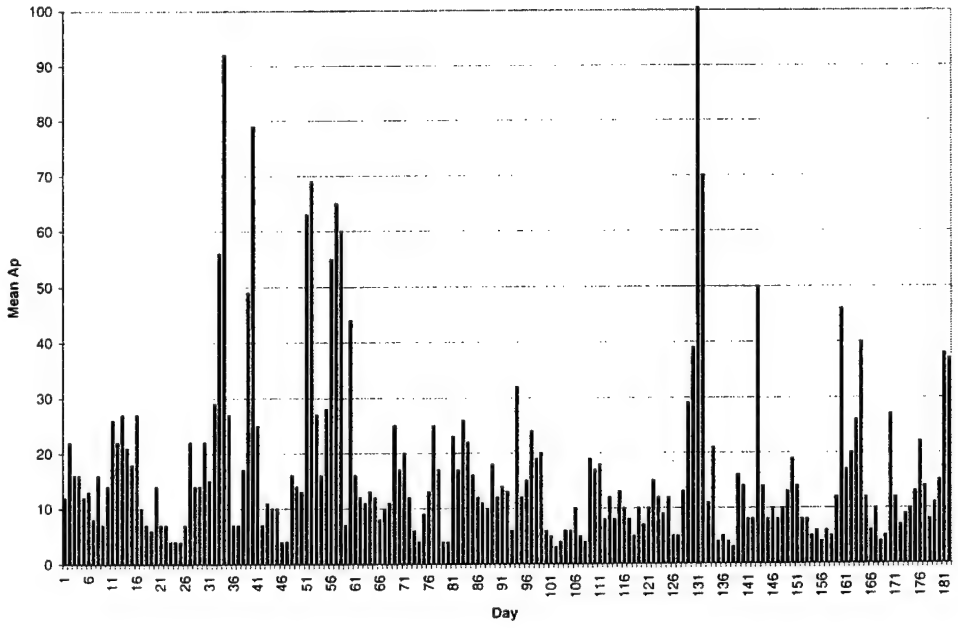


Figure 4.4: Average Planetary Amplitude (Ap), Jan – Jun 1992

A typical indicator of the relative strength of geomagnetic disturbance is how many daily Ap values exceed 40. From comparison of these two plots, we can conclude that our two-month interval is not particularly perturbed, but features more variation than some quieter intervals

We also present the real and predicted Schatten values of the solar $F_{10.7}$ flux:

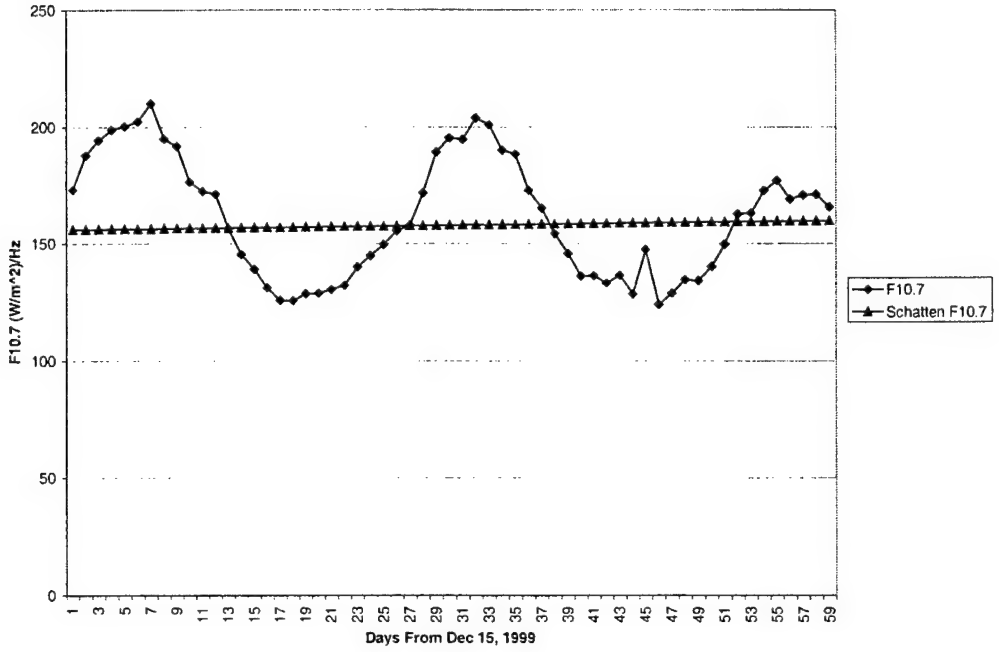


Figure 4.5: Daily 10.7 cm Solar Flux, Dec 15, 1999 - Feb 11, 2000

The $F_{10.7}$ values range over the course of the eleven-year solar cycle from a minimum of approximately 65 to a maximum of 200 [56]. Therefore, we see that our interval occurs during a relatively "hot" epoch, which means that the atmosphere will be at higher temperatures and more dense than during "cold" epochs. Although these conditions will cause LEO objects to decay more rapidly, our density correction process should actually function better because the effect of drag on satellites can be more easily detected.

The truth orbits will be generated from a TLE file dated Dec 14, 1999, and taken from the JPL FTP site [49]. The TLE file was limited to objects with apogees and perigees in the range previously described in Section 4.1.1. A number of the 454 objects

were removed from consideration because they were known to be debris or exhibited rapid decay behavior, leaving us with 335 objects to be used for atmospheric correction. These objects have the following distribution of ballistic factors:

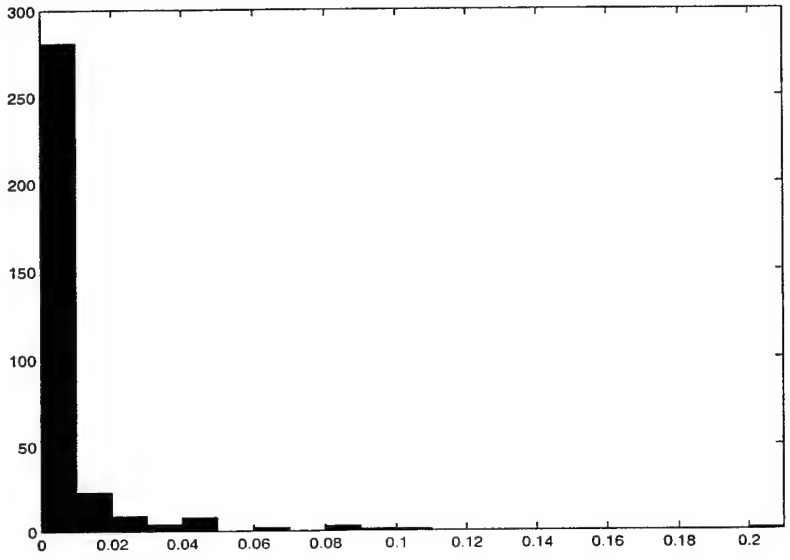


Figure 4.6: Histogram of Ballistic Factors

As can be seen from the above figure, most of the ballistic factors are in the range of [0,0.01]. A histogram of starting perigee heights can also be plotted:

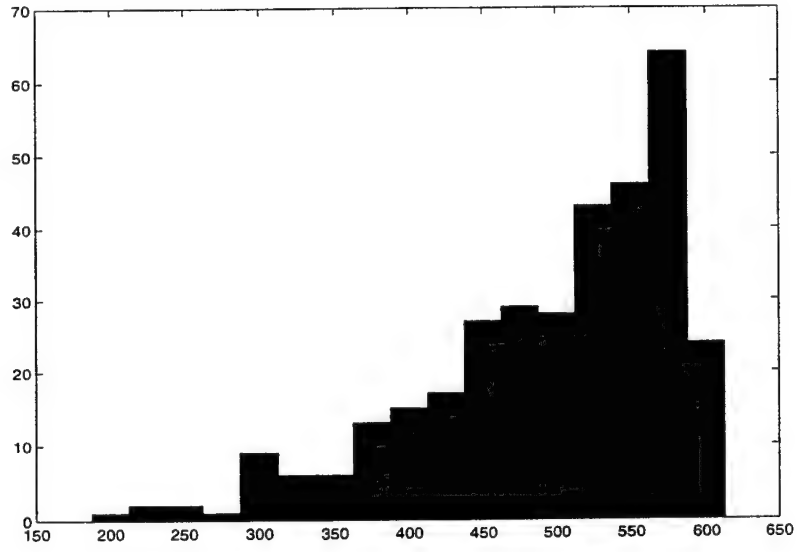


Figure 4.7: Histogram of Perigee Heights in Kilometers

The perigee heights are seen to fall mostly between 300 and 600 km, which is in approximate agreement with the results presented by Nazarenko in DFY 97 Stage 3 of his report [1]. It appears that we are using more high-altitude objects than Nazarenko, but this can be explained by the fact that we have more objects total in the database. If we removed some of the high-altitude objects, the histogram would undoubtedly be closer in appearance to what appears in Ref. [1].

To improve computational speed, the truth file will be generated using the JGM2 gravity model truncated to degree and order of 4x4. This simplification will still allow us to capture the approximate real-world evolution of the orbits, and will not introduce any error in the density correction process as long as the fit gravity model is also JGM2 4x4. Solar radiation pressure effects will be included, with the solar reflectivity constant (C_R) assigned the default value of 1.2. Third-body effects such as solar and lunar perturbations will also be taken into account.

4.3.2 Correction of an Inaccurate Density Model

This test case will validate the effectiveness of the density correction process in capturing short-term variations that are missing in a given density model. All objects are assumed to be standard, i.e. the calculation of variations will have access to all true ballistic factors. The truth model will remain JR-71, but the fit model is JR-71 with "smoothed" values which were originally taken from a Schatten predict file. The smoothed value of Ap over the 58-day interval is approximately equal to 15, while the smoothed $F_{10.7}$ slowly increases from 154 to 158. Plots of these values can be seen in Figure 4.3 and Figure 4.5 above.

Other perturbations and force models are the same as in the first test case. Tests will be executed with noise added to the simulated range, azimuth, and elevation observations, where the noise characteristics are specific to each ground station and can be found in the SLAD file described earlier. Two measures of quality will be used: 1) comparison of the actual density in the truth model to the modeled density plus corrections, and 2) execution of differential correction runs for low, medium, and high-altitude objects in the database. The observations will be fit to the smoothed model without corrections, and the radial, cross-track, and along-track errors will be plotted for

a given fit and predict interval. Then, the same observations will be fit to the atmospheric model with corrections and the resulting fit and predict plots and statistics can be compared to the first case.

One more validation will take place in this phase of testing. The number of objects available to the density correction process will be reduced from the full 333 to a number near 200, and the test cases will be repeated. This test should allow us to observe the dependence of density correction on the number of calibration satellites available.

4.3.3 Correction of an Inaccurate Model With Forecasting

This test case will focus on validation of the forecasting algorithms presented in Section 2.3. Forecasted density variation coefficients for two different epochs will be calculated and compared with "true" coefficients. The accuracy of forecasting will be determined with respect to length of prediction interval. Calculation of the deterministic and random component of each signal will be validated.

Chapter 5 Results

All test cases follow the general pattern and are executed under the conditions described in Section 4.3.

5.1 End-to-End Software Validation

The purpose of this test is to ensure that all the pieces of the simulation connect properly, and to show that the density variations are equal to zero if our fit model is equal to the truth model. This test case was executed under the conditions described in Section 4.3.1. The density variations were only calculated for fifteen days, as this was thought to be a sufficient amount of time to determine if the algorithms are functioning properly. Figures 5.1 and 5.2 illustrate the progression of the density variation model coefficients b_1 and b_2 over time:

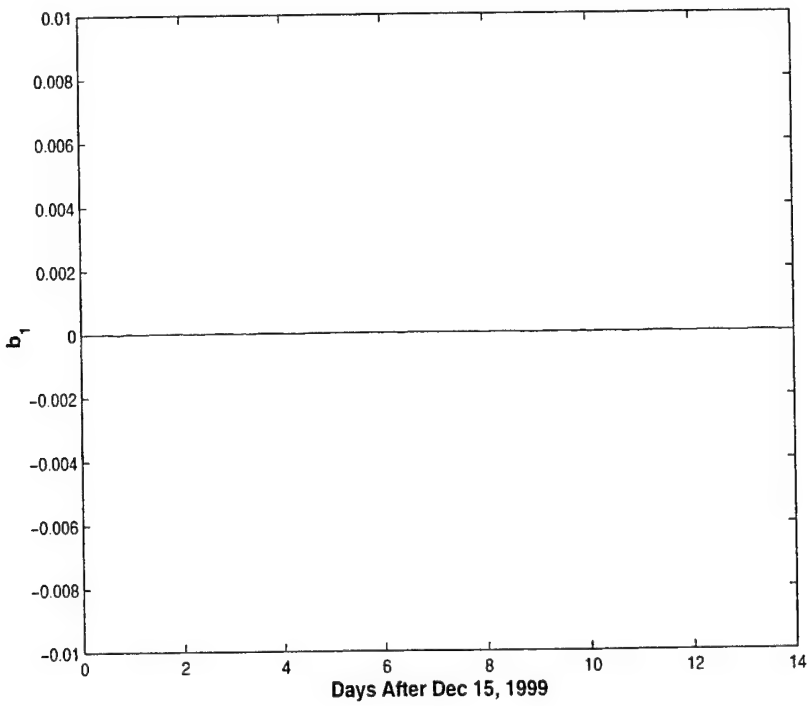


Figure 5.1: Density Variation Coefficient b_1, No Mismodeling

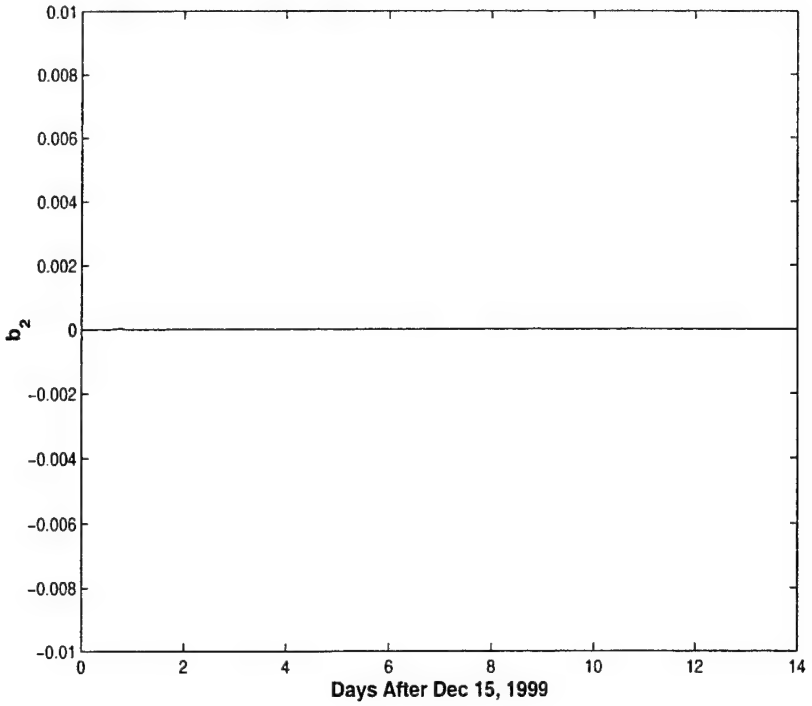


Figure 5.2: Density Variation Coefficient b_2, No Mismodeling

As expected, the variation coefficients are essentially equal to zero over the entire time interval. The mean value of b_1 = -2.6487x10^{-07} with a maximum of 6.4747x10^{-06}, and the mean of b_2 = 2.8295x10^{-07} with a maximum of 3.2653x10^{-05}. These results indicate that the differential correction process is not introducing any significant error into the calculation of density variations. The results also show that all pieces of software and associated interfaces shown in Figure 4.1 are properly connected. As these results do not shed any light on any other aspects of density correction, we shall move on to the Schatten mismodeling cases.

5.2 Correction of an Inaccurate Density Model

For the second series of test cases, described in Section 4.3.2, the objective is to determine if the density variations can capture the difference between a truth density model and a smoothed Schatten fit model. This determination can be made through

direct examination of the density coefficients, comparison of actual density values at various altitudes, and the execution of differential correction for satellites in test orbits.

5.2.1 Calculation and Analysis of Density Variation Coefficients

Density corrections were calculated using the same objects as the first test case but with noisy measurements and mismodeling of the atmosphere. Density variation models were produced every three hours for the entire 55-day interval. Time histories of the model coefficients are given in Figure 5.3 and Figure 5.4 below:

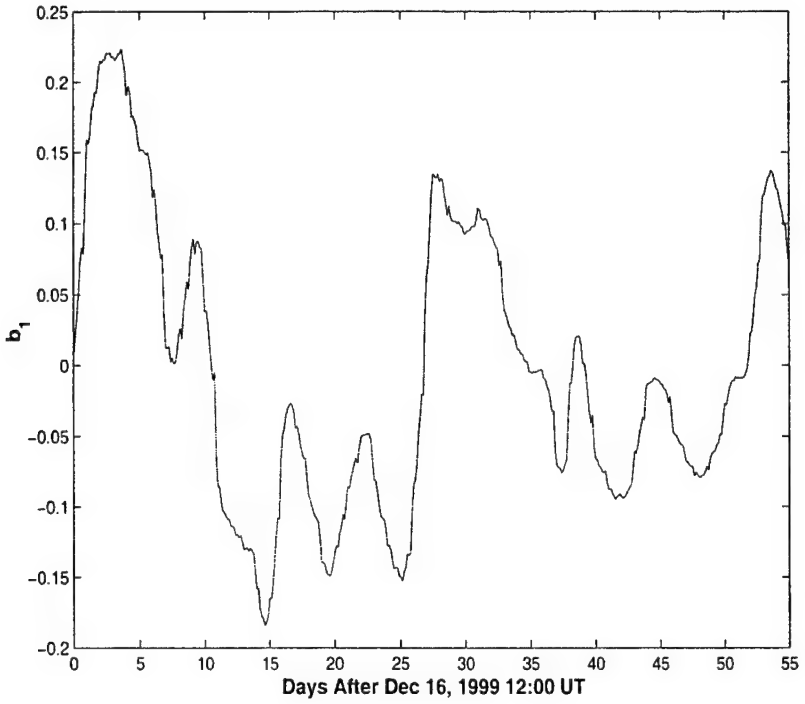


Figure 5.3: Density Variation Coefficient b_1, Schatten Mismodeling and Noise

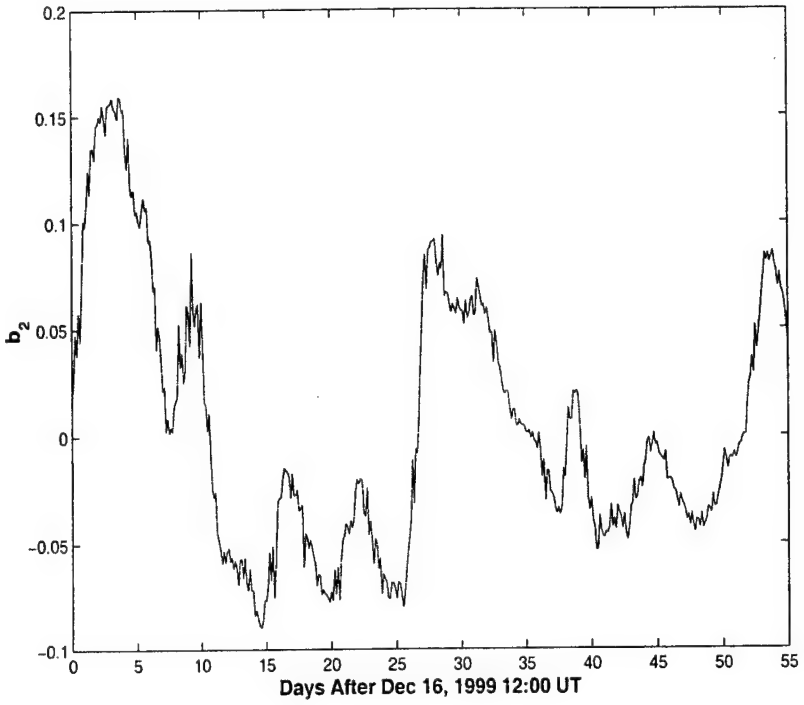


Figure 5.4: Density Variation Coefficient b_2, Schatten Mismodeling and Noise

Note that the density variations begin a day and a half after the beginning of the simulation time period. This is because the first estimated ballistic factors are attributed to the midpoint of the first three-day fit interval. When compared to the general trend of atmospheric conditions given in Figure 4.3, we see that the major period of the variations is approximately 25-26 days, which matches the period of the fluctuations in $F_{10.7}$ values. We also observe short-term spikes in the density variations, which seem to correspond to rapid changes in geomagnetic conditions as measured by the Ap index in Figure 4.5. This does not necessarily mean that the density corrections are applied with appropriate magnitude and/or phase; other tests will be necessary to make this determination.

5.2.2 Comparison of Actual Density Values

In some sense, the true measure of how well the density correction process performs is found in an examination of actual values of density at various altitudes and times. We computed the actual density encountered by a simulated spacecraft in a

circular, equatorial orbit at 200 km, 400 km, and 600 km over the entire 55-day span. Three density histories were computed at each altitude: the truth density, the uncorrected Schatten density, and the corrected Schatten density. The densities were computed every 60 seconds over two time intervals respectively chosen for their relatively quiet and perturbed geomagnetic conditions. These intervals were used for the fit and predict intervals for the DC test cases which follow later in this section.

5.2.2.1 Quiet Epoch

The quiet interval extends from Dec 17, 1999 to Dec 25, 1999. A detailed picture of the quiet geomagnetic conditions is given in the following figure:

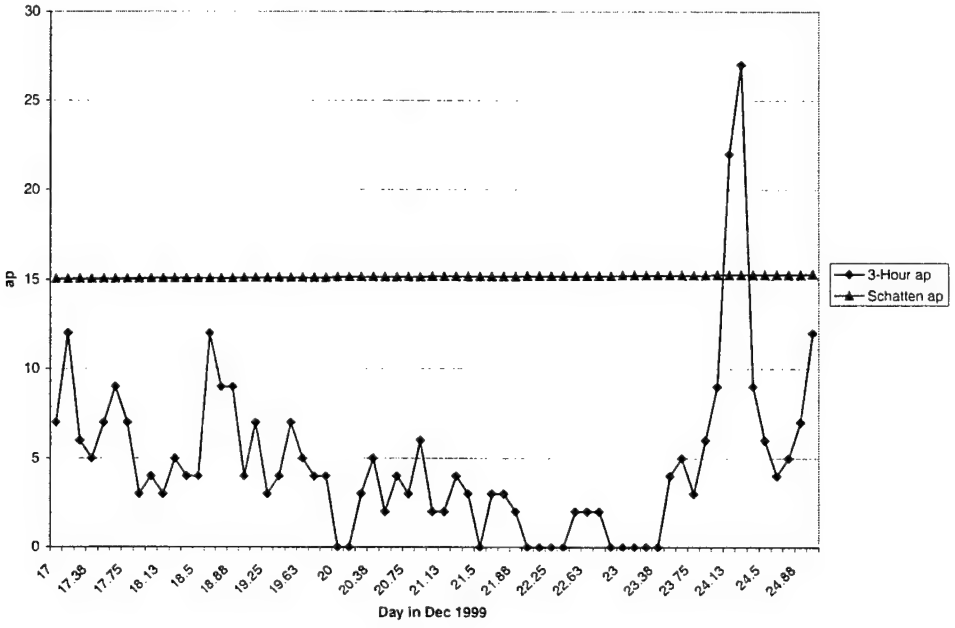


Figure 5.5: Three-Hour Values of ap for Quiet Interval

Note that ap refers to the three-hour measurement, while Ap is equal to the one-day average of the eight ap values. The relative error in percent of the Schatten density and corrected Schatten density for the quiet epoch at 200 km is presented in Figure 5.6 below:

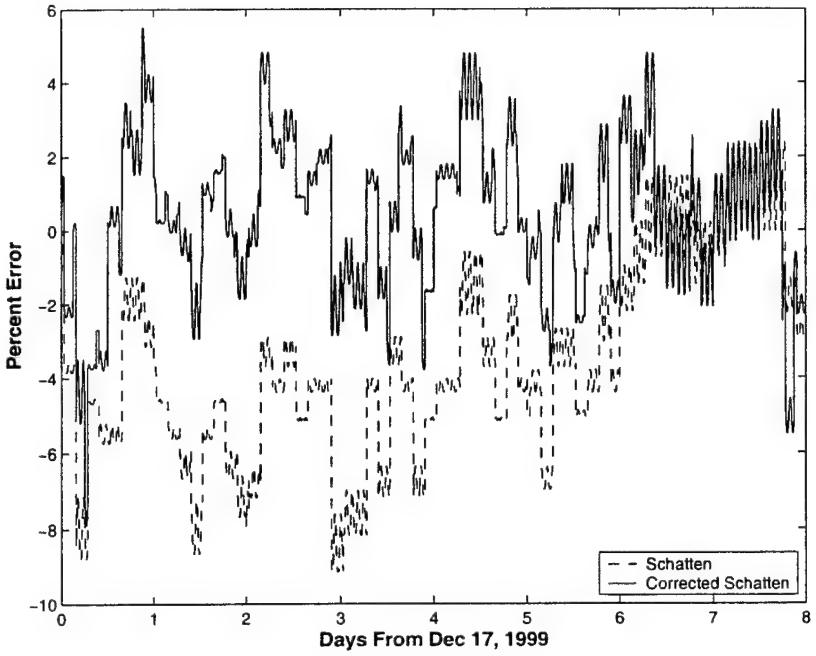


Figure 5.6: Relative Error in Uncorrected and Corrected Density at 200 km, Quiet Epoch

We observe two trends in the Schatten density error: a short-term three-hour component corresponding to errors in geomagnetic conditions, and an overall bias most likely due to inaccurate $F_{10.7}$ values. The average Schatten error over this eight-day interval is equal to –3.51%, and the standard deviation is equal to 2.71%. The density correction process appears to have removed most of the bias and some of the short-term variation: the mean corrected error = 0.38%, and the standard deviation is 2.13%. Significant short-term variations remain in the corrected density, which indicates that the correction process is not able to capture all of the effects due to unmodeled geomagnetic disturbances. This could be due to a number of reasons: 1) the differences between true ap and Schatten ap are relatively small for this interval; 2) the fit interval used in estimation of ballistic factors is too long; and 3) the observation data rates are not high enough.

The figures below present relative density errors at 400 km and 600 km:

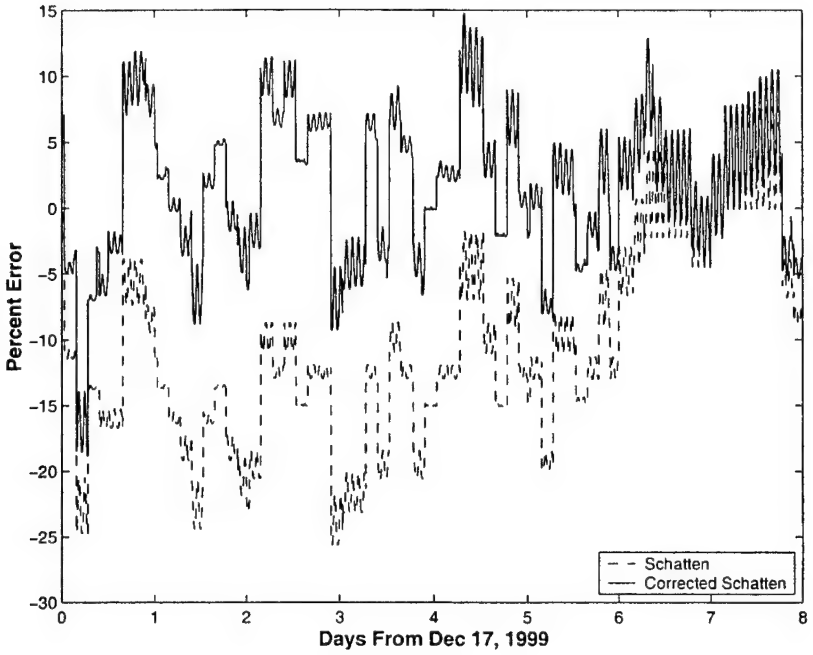


Figure 5.7: Relative Error in Uncorrected and Corrected Density at 400 km, Quiet Epoch

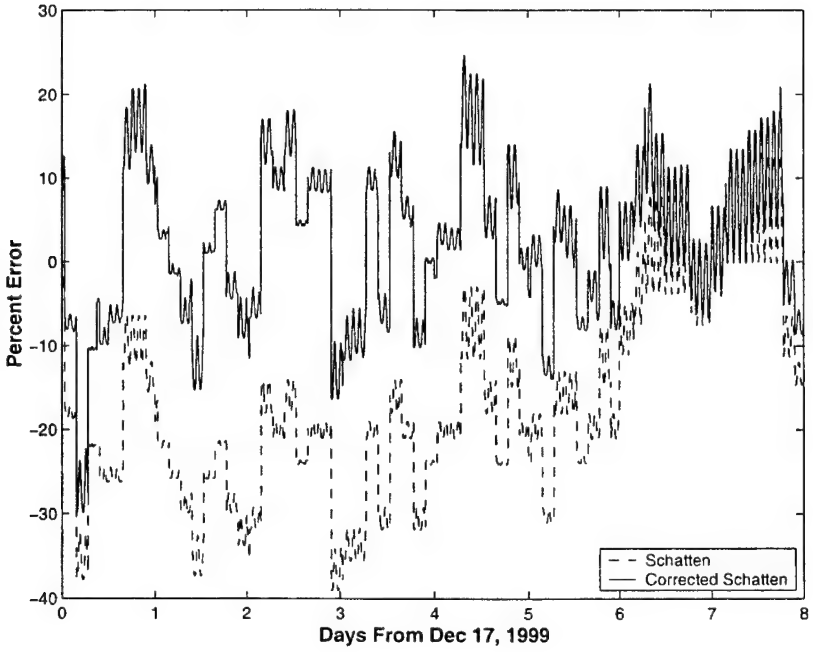


Figure 5.8: Relative Error in Uncorrected and Corrected Density at 600 km, Quiet Epoch

These figures are almost identical to the 200 km case except in scale, with the uncorrected density error reaching almost 40% in the 600 km case. This trend is to be expected for higher altitude regimes, where relative density fluctuations are considerably greater than for lower altitudes.

5.2.2.2 Perturbed Epoch

The perturbed interval covers the eight days from Dec 28, 1999 to Jan 5, 2000, with geomagnetic conditions detailed in Figure 5.9:

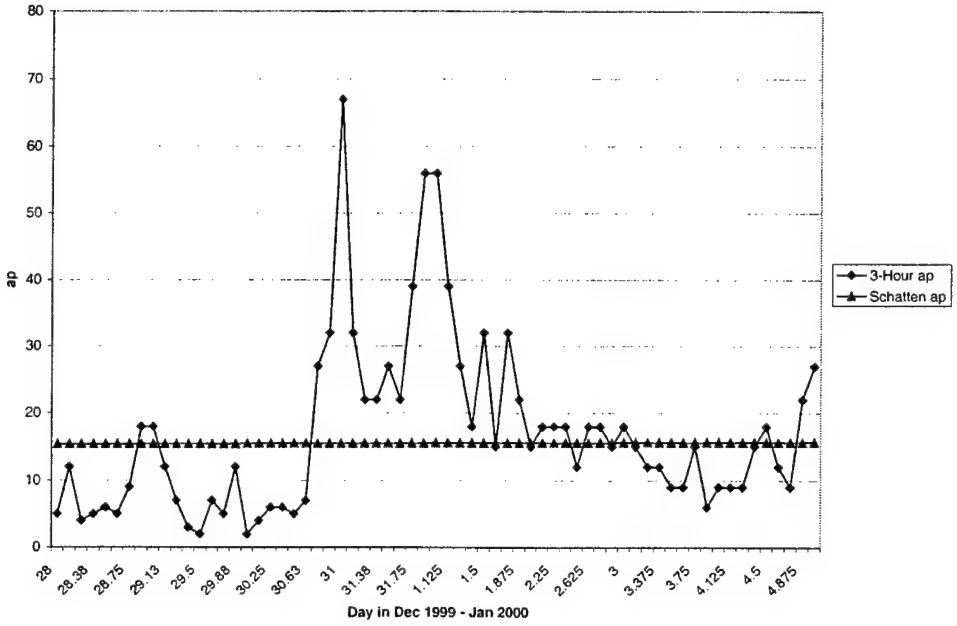


Figure 5.9: Three-Hour Values of ap for Perturbed Interval

The perturbed interval is seen to contain a significant spike in ap, which should be reflected in the density error plots. It should be noted that the JR-71 model assumes a 6.7-hour lag between the time of change in ap value and its actual effect on density.

Figures 5.10-5.12 illustrate density errors at 200, 400, and 600 km:

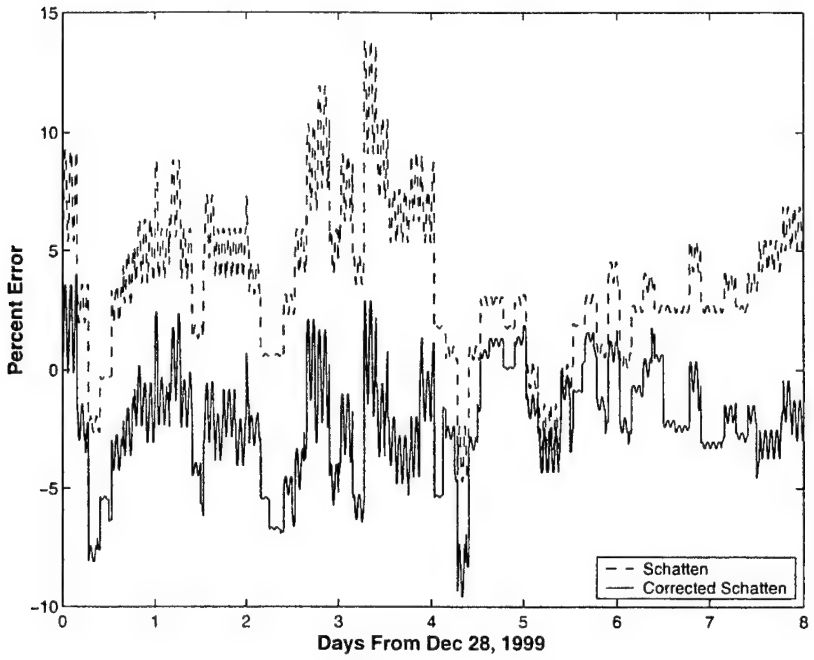


Figure 5.10: Relative Error in Uncorrected and Corrected Density at 200 km, Perturbed Epoch

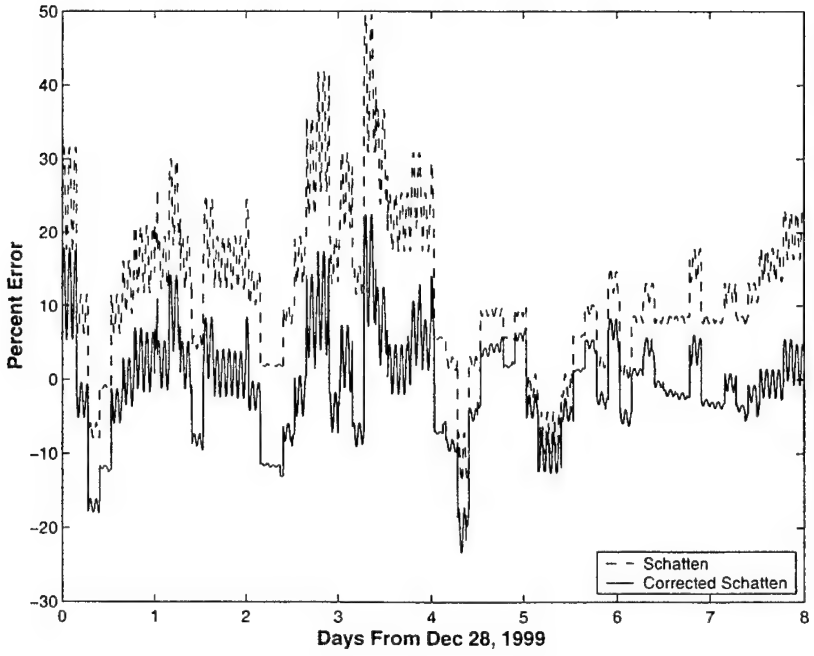


Figure 5.11: Relative Error in Uncorrected and Corrected Density at 400 km, Perturbed Epoch

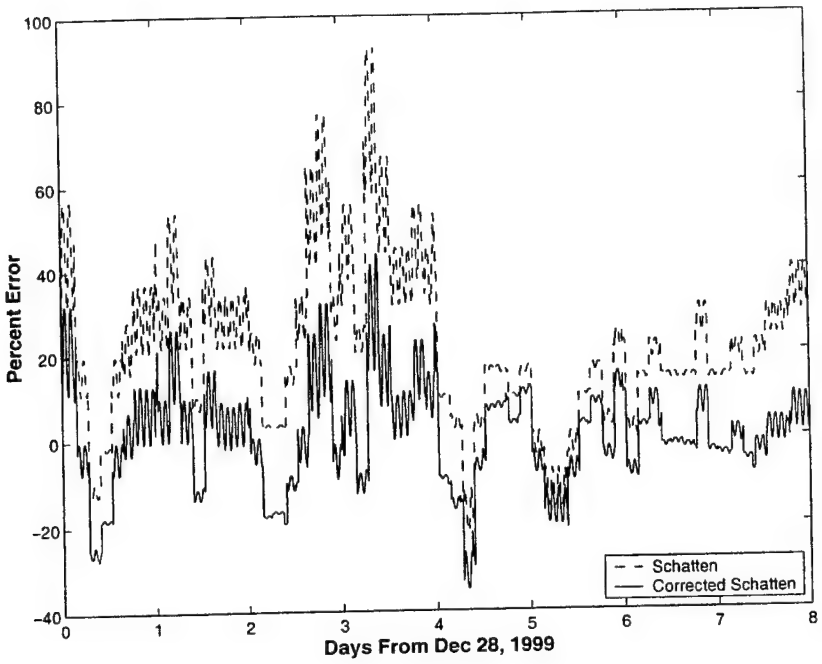


Figure 5.12: Relative Error in Uncorrected and Corrected Density at 600 km, Perturbed Epoch

There is indeed a significant spike in error corresponding to the unmodeled jump in ap on the third day. The corrected density does not completely remove the spike, but at a minimum shifts it down to a more appropriate regime. We again observe that short-term errors remain in the corrected density, but the biases have been essentially removed. Table 5.1 summarizes the error statistics for both the quiet and perturbed epochs:

Table 5.1: Density Error Statistics

Altitude (km)	Case	Quiet Epoch Error (%)		Perturbed Epoch Error (%)	
		Mean	Std. Dev.	Mean	Std. Dev.
200	Schatten	-3.51	2.71	3.65	3.11
	Corrected	0.38	2.13	-2.17	2.32
400	Schatten	-10.26	7.83	12.07	10.44
	Corrected	1.58	5.66	-0.49	7.07
600	Schatten	-16.33	12.35	21.43	18.71
	Corrected	2.06	9.46	0.80	11.95

In all cases, the corrected mean error is reduced to 2% or less, and the corrected standard deviation is reduced by 21 – 36%.

5.2.3 Quiet Epoch DC Test Cases

We next executed a number of GTDS DC test cases in each epoch to determine if corrections to a density model could improve orbital fits and predictions. The first DC test cases were run for four objects in the density calibration database. The objects were chosen to have varying perigee heights, eccentricities, inclinations, and ballistic factors in order to thoroughly test the density correction process. Orbital elements and ballistic factors for each object are given below:

Table 5.2: Orbital Elements and Ballistic Factors for Test DC Objects

NSSC #	h_{per} (km)	e	i (deg)	k
09854	303.2	0.00116	80.87	0.005241
25013	398.5	0.00521	44.94	0.002228
17769	568.9	0.01245	98.66	0.08469
25947	232.7	0.03358	51.77	0.005988

We will begin with NSSC# 09854, which in some sense is the "easiest" test case in that it is in a moderately low-altitude circular orbit with an average ballistic factor.

Initially, a three-day window from Dec 17 – Dec 20, 1999 was used as the quiet fit interval. However, not all of the test objects had enough observations during this interval to allow the DC program to converge on a viable solution without density correction. We analyzed the observation frequencies for our simulated calibration database, and compared the results to average Air Force Space Command observation frequencies. Using a tracking schedule such that each ground station tracked all visible objects for 6 hours each day, the average number of simulated observations per object per day is approximately 30. (One observation is defined as a given set of measurements with a particular epoch). Actual Air Force Space Command data were also analyzed, and it was found that the average number of observations works out to be approximately 36 observations per object per day. Thus, it is possible that our data is a bit too sparse in some cases (such as the very low-altitude objects) to allow for good convergence, but should on the average be a conservative approximation of real-world conditions.

With our tracking schedule validated, we decided to extend the fit interval to five days instead of trying to re-simulate new data. The data were fit to the smoothed Schatten density model without corrections, and the estimated state vector was then propagated over the five-day fit interval and a three-day predict interval again using the smoothed Schatten model. We then executed the same test cases using the corrected density model in both fit and prediction intervals. This test model is perhaps not too realistic with respect to real-world tracking, since we have access to the corrected model in the future as well as the past. However, such a test scheme is more similar to real-world problems such as post-processing of observation data to estimate spacecraft characteristics such as mass.

Errors in the radial, cross-track, and along-track directions were computed for spacecraft position and velocity. We can see from Figure 5.13 below that the DC program has some trouble fitting the observations to the inaccurate density model. A maximum error of 1505 m and 1634 mm/sec and RMS error of 586 m and 662 mm/sec are observed.

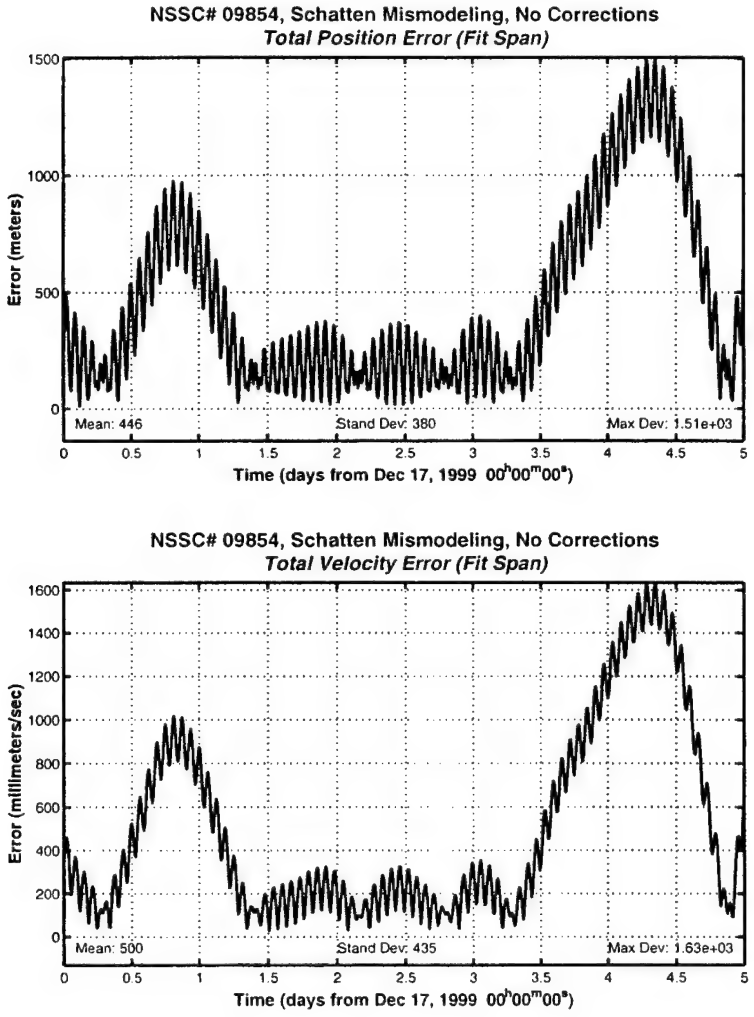


Figure 5.13: Fit Error for NSSC# 09854, Quiet Epoch, Schatten Mismodeling, No Corrections

When the corrections are applied to the density model, however, the fit is much better. The max state errors are 808 m and 866 mm/sec and the RMS state errors are 302 m and 334 mm/sec, as seen in Figure 5.14 below.

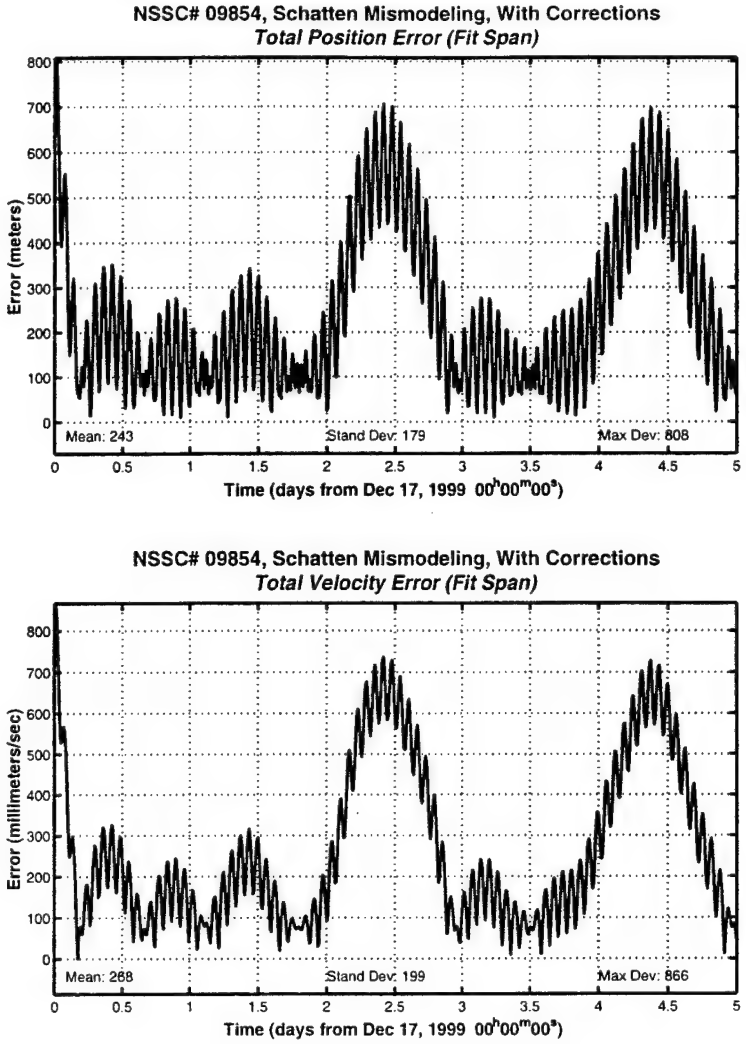


Figure 5.14: Fit Error for NSSC# 09854, Quiet Epoch, Schatten Mismodeling, With Corrections

The efficacy of the process is displayed in a much better fashion, however, in the three-day predict interval. The DC program estimates a drag coefficient that best fits the data in the fit interval, but if atmospheric conditions change in the predict interval, the estimated drag coefficient is no longer appropriate. The result is that drag effects in the predict interval cause quadratically increasing errors in just a few days, as can be seen in the following Figure 5.15. The max state errors grow to 55.4 km and 63.8 m/sec in 72 hours.

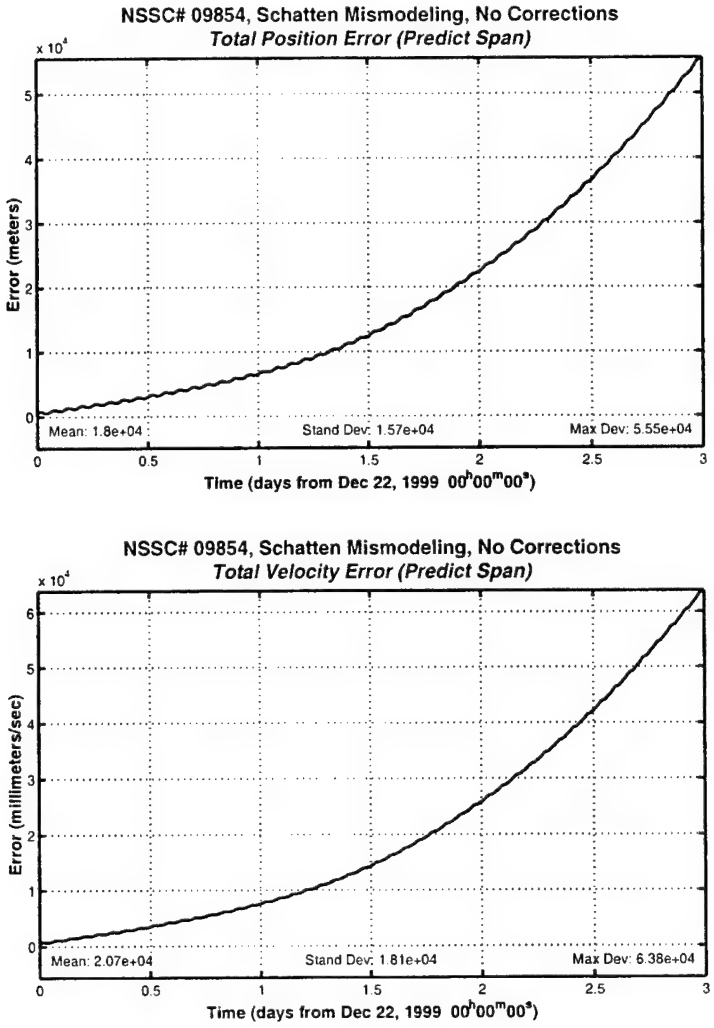


Figure 5.15: Predict Error for NSSC# 09854, Quiet Epoch, Schatten Mismodeling, No Corrections

When the density corrections are applied, the error is reduced by more than an order of magnitude: the max errors over the three-day predict span are 3.27 km and 3.71 m/sec (Figure 5.16). The error reaches the 1-km level after approximately 32 hours, as opposed to just a few hours for the uncorrected density case. This result indicates that we are obtaining a better-estimated drag coefficient (in the form of ρ_1) in the fit interval. The quality of estimated drag coefficients is discussed in more detail later in this section.

For plots of the position and velocity error in the radial, cross-track, and along-track directions, please refer to Figures A.1-4 in Appendix A.

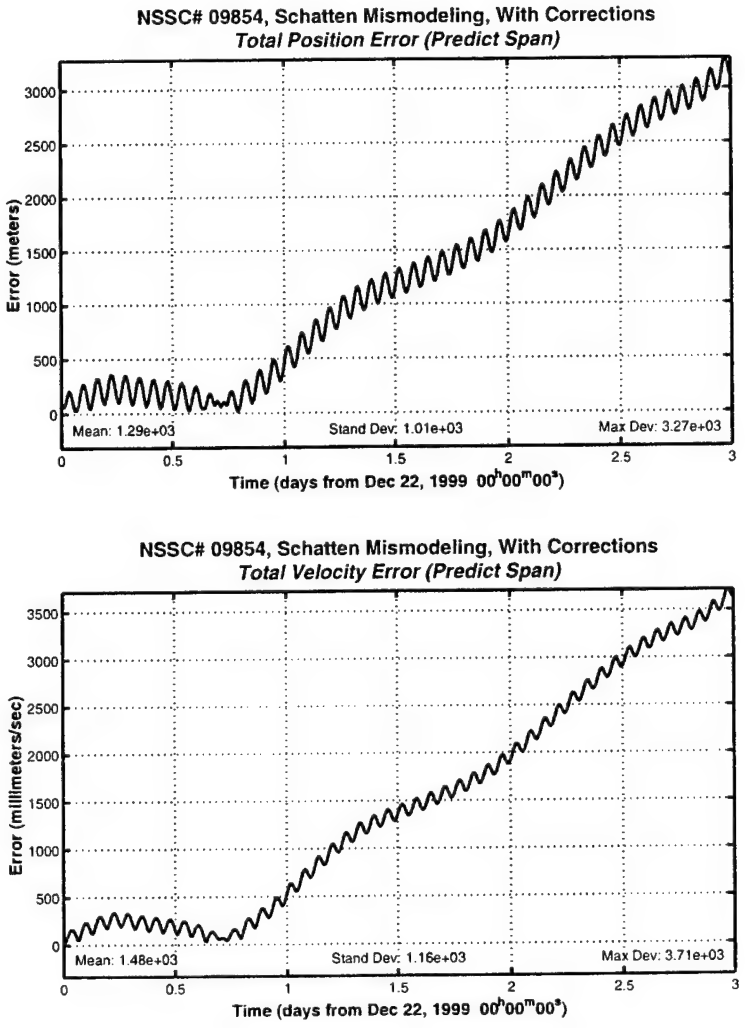


Figure 5.16: Predict Error for NSSC# 09854, Quiet Epoch, Schatten Mismodeling, With Corrections

The position error characteristics for each object are compiled in Table 5.3 below. The maximum and root-mean-square (RMS) error in the radial, cross-track, and along-track directions is presented for each of the four test objects, with the density model uncorrected and corrected, for the fit interval and predict intervals. All numbers are given in units of meters.

Table 5.3: Position Error Characteristics for Quiet Epoch Test Cases

NSSC#	Uncorr/ Corr	Radial Error (m)		Cross-Trk Error (m)		Along-Trk Error (m)		Total Error (m)	
Fit Span: Dec 17 – Dec 22, 1999									
		Max	**RMS**	**Max**	**RMS**	**Max**	**RMS**	**Max**	**RMS**
09854	Uncorr	112.9	65.04	33.21	20.30	1504.	582.4	1505.	586.3
	Corr	96.47	48.26	11.17	7.607	807.5	298.0	808.2	301.9
25013	Uncorr	32.74	21.12	78.40	52.43	212.4	80.27	212.5	98.14
	Corr	14.88	7.777	34.59	22.69	94.27	43.40	94.85	49.58
17769	Uncorr	67.09	40.70	33.55	21.03	713.7	192.7	713.8	198.1
	Corr	15.49	4.706	36.92	23.97	243.8	86.02	246.1	89.39
25947	Uncorr	95.00	44.23	21.84	14.81	2215.	511.1	2216.	513.1
	Corr	406.3	235.8	242.7	156.1	2511.	741.8	2514.	793.7
Predict Span: Dec 22 – Dec 25, 1999									
09854	Uncorr	612.4	223.1	33.70	22.21	55461	23918	55462	23919
	Corr	77.44	45.76	11.39	7.884	3274.	1635.	3274.	1636.
25013	Uncorr	85.10	31.19	68.77	45.97	6213.	2776.	6213.	2776.
	Corr	11.60	7.387	29.42	19.93	133.8	67.02	133.9	70.30
17769	Uncorr	253.2	69.25	41.25	26.32	16345	7367.	16347	7367.
	Corr	21.23	6.525	40.12	27.27	1780.	891.4	1780.	891.9
25947	Uncorr	2416.	629.8	26.64	10.01	57799	25607	57850	25614
	Corr	718.5	330.7	200.1	135.5	11748	5259.	11749	5271.

We immediately notice that the position error is dominated by the along-track component, which is typical for drag-pertubed orbit determination problems.

The next test cases were executed for NSSC# 25013, which tests the process at a slightly higher altitude and eccentricity. As we see from Table 5.3 above and from Figure A.5 in Appendix A, the improvements are even more dramatic. The total fit error is more than cut in half, and the total RMS predict error decreases from 6213 m to 133.9 m, a 46-fold improvement.

At higher altitudes, the density correction process was again shown to be effective. Total error plots for NSSC# 17769 in the fit and predict intervals with and without corrections are included in Figure A.6. The total RMS error for NSSC# 17769 is cut from 7367 m to 891.9 m, or by about 8 times. The improvements are not as drastic as for the lower-altitude cases, which is what we would expect since objects at higher altitudes are not as drag-perturbed.

The final test case, using measurement from NSSC# 25947 and presented in Figure A.7, is the most challenging due to the object's high eccentricity and relatively low perigee height. The DC program had a great deal of trouble fitting the observations to the uncorrected density model, as is evidenced by the 57.9 km max predict error after

three days. With corrections applied, the statistics are considerably better: 11.7 km max predict error, or about a five-fold improvement. It is interesting to note, however, that the fit interval error with corrections is actually slightly worse than without corrections. We can explain this phenomenon using statistics in Table 5.4:

Table 5.4: Fit Statistics for Quiet Epoch Test Cases

NSSC #	Uncorr/ Corr	ρ_1	σ_{ρ_1}	σ_a (m)	A-priori Accuracy
09854	Uncorr	0.138	0.568E-4	0.850	100 m
	Corr	-0.016	0.401E-4	0.106	100 m
25013	Uncorr	0.370	0.670E-3	0.214	100 m
	Corr	-0.012	0.379E-3	0.173	100 m
17769	Uncorr	0.208	0.138E-2	0.451	10 m
	Corr	-0.017	0.114E-2	0.464	10 m
25947	Uncorr	0.100	0.915E-3	0.427	100 m
	Corr	-0.012	0.760E-3	0.394	10 m

This table presents the converged values of ρ_1 and associated standard deviation, and the standard deviation of the semi-major axis. Also shown is the a-priori state accuracy required to achieve convergence. We see that fitting observations with density corrections results in smaller values of ρ_1, indicating that the corrected density model is in fact closely approaching the truth density model. The standard deviation of ρ_1 also improves in every case, although more significantly for the low and middle-altitude cases. For the high-altitude and eccentric cases, we see that the quality of the fit as measured by σ_a does not greatly improve or even degrades going from corrections to no corrections. However, because we have more accurately estimated the true ballistic factor, the error in the predict spans is not nearly as great.

The last column of the table presents the necessary a-priori accuracy in the state vector (position and velocity) to allow the DC program to obtain a valid solution from the observations. The number in meters corresponds with the required velocity accuracy in units of mm/sec. The required accuracy is the same in all cases except for NSSC# 25947:

for this object, the corrected density model actually required more a-priori accuracy to converge. This corresponds with our earlier observation that the fit with corrections is worse for this object, which may be due to poor observability of the drag coefficient with relatively sparse observations.

5.2.4 Perturbed Epoch Test Cases

The same test cases were initially executed fitting five days of data from Dec 28, 1999 to Jan 2, 2000. The same data rates as for the first test cases were used, but significant convergence problems arose. Even with 7 or 8-digit apriori state accuracy, uncorrected fits caused the DC program to exceed the allowable number of iterations or to stop due to numerical instability. The corrected fits,however, were able to converge with sparse data in all cases. Plots of the total error in the predict and fit intervals for NSSC# 09854 are presented in Figure 5.17 below:

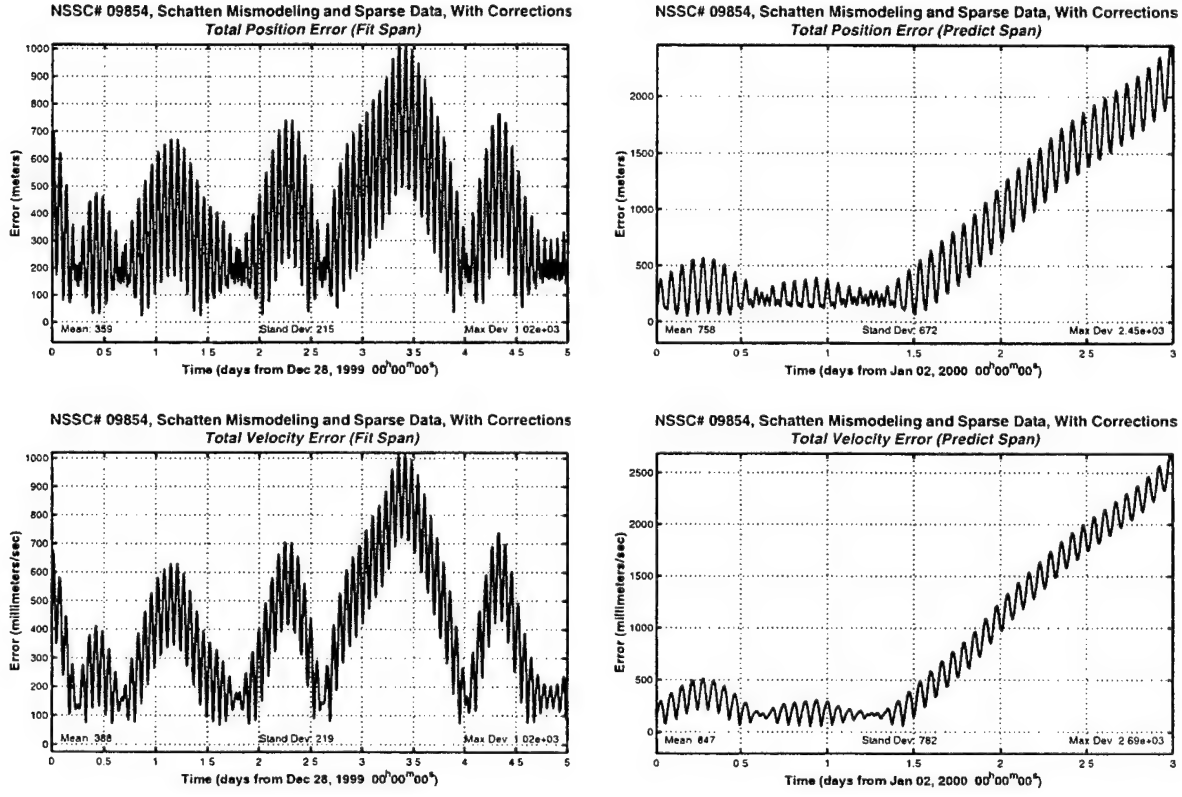


Figure 5.17: Fit and Predict Errors for NSSC# 09854, Perturbed Epoch, Schatten Mismodeling, With Corrections

For plots of radial, cross-track, and along-track error, refer to Appendix A Figures A.8 and A.9 We can observe a maximum in the fit error on the fourth day corresponding to the spike in ap values shown in Figure 5.9 earlier. Additional results for these sparse-data fits are presented in Table 5.5 which follows and in Figures A.10-A.11.

Table 5.5: Position Error Characteristics for Perturbed Epoch Test Cases with Sparse Data

NSSC #	Uncorr/ Corr	Radial Error (m)		Cross-Trk Error (m)		Along-Trk Error (m)		Total Error (m)	
Fit Span: Dec 28, 1999 – Jan 2, 2000									
		Max	**RMS**	**Max**	**RMS**	**Max**	**RMS**	**Max**	**RMS**
09854	Uncorr	Did not converge							
	Corr	150.2	93.05	75.51	45.62	1014.	405.7	1015.	418.5
25013	Uncorr	Did not converge							
	Corr	21.60	13.99	8.715	5.711	206.7	66.45	206.7	68.12
17769	Uncorr	55.35	34.60	104.0	64.13	452.2	184.6	461.9	198.3
	Corr	170.3	116.8	37.89	26.58	828.0	445.5	828.6	461.3
25947	Uncorr	Did not converge							
	Corr	281.0	183.0	24.83	16.53	1641.	748.9	1641.	771.0
Predict Span: Jan 2, 2000 – Jan 5, 2000									
09854	Uncorr	Did not converge							
	Corr	142.8	93.39	92.07	59.60	2450.	1006.	2450.	1012.
25013	Uncorr	Did not converge							
	Corr	22.00	13.83	9.207	6.368	676.7	336.7	676.8	337.0
17769	Uncorr	184.7	77.04	119.2	79.44	11111	5832.	11112	5833.
	Corr	166.4	108.2	38.63	26.97	3185.	1750.	3185.	1753.
25947	Uncorr	Did not converge							
	Corr	669.5	270.8	33.44	19.00	19536	9663.	19536	9667.

We can see from these results that the density correction process allows us to maintain operability even for very low data rates. The DC program was able to converge on a solution using the uncorrected density model for NSSC# 25947, but the corrected solution seems to be better in terms of predict errors. This is due to the fact that the DC runs were able to capture the ρ_1 values very well:

Table 5.6: Fit Statistics for Perturbed Epoch Test Cases with Sparse Data

NSSC #	Uncorr/ Corr	ρ_1	σ_{ρ_1}	σ_a (m)	A-priori Accuracy
09854	Uncorr	Did not converge			
	Corr	0.017	0.166E-3	0.518	10 m
25013	Uncorr	Did not converge			
	Corr	0.002	0.523E-3	0.101	1 m
17769	Uncorr	-0.234	0.704E-3	0.097	10 m
	Corr	-0.004	0.853E-3	0.090	100 m
25947	Uncorr	Did not converge			
	Corr	0.032	0.239E-2	0.579	100 m

In order to obtain comparable results for corrected and uncorrected cases, we regenerated simulated observations with identical characteristics but increased the data rate by a factor of six to 180 obs/object/day (on average). Accordingly, the fit interval was reduced back to three days. The predict interval remained at three days, and the density correction model remained the same. Table 5.7 presents the error characteristics:

Table 5.7: Position Error Characteristics Perturbed Hot Epoch Test Cases with Dense Data

NSSC #	Uncorr/ Corr	Radial Error		Cross-Trk Error		Along-Trk Error		Total Error	
Fit Span: Dec 28 – Dec 31, 1999									
		Max	**RMS**	**Max**	**RMS**	**Max**	**RMS**	**Max**	**RMS**
09854	Uncorr	143.0	90.51	118.7	77.71	888.6	343.0	889.4	363.0
	Corr	310.6	211.8	103.6	67.78	1493.	694.0	1496.	728.3
25013	Uncorr	19.43	8.282	32.39	22.13	2334.	943.1	2334.	943.4
	Corr	22.20	14.04	12.03	8.198	265.4	72.49	265.5	74.26
17769	Uncorr	99.38	65.28	14.34	9.821	527.4	243.9	527.7	252.7
	Corr	88.22	58.68	13.10	8.950	573.2	231.4	573.5	238.8
25947	Uncorr	566.5	116.0	66.01	42.43	11187	2252.	11188	2255.
	Corr	165.0	104.0	85.64	56.94	1690.	733.1	1690.	742.5
Predict Span: Dec 31, 1999 – Jan 3, 2000									
09854	Uncorr	196.1	100.9	144.6	93.06	12292	8561.	12292	8563.
	Corr	328.9	212.7	125.5	81.72	7178.	3457.	7178.	3464.
25013	Uncorr	37.72	16.28	32.68	23.05	5642.	3074.	5642.	3074.
	Corr	23.23	14.07	12.86	8.799	622.1	349.8	622.1	350.2
17769	Uncorr	110.3	69.18	13.47	9.100	1142.	625.6	1142.	629.2
	Corr	83.48	55.10	13.14	8.939	2149.	1014.	2149.	1015.
25947	Uncorr	532.9	261.4	82.32	52.49	17214	12061	17214	12064
	Corr	356.8	135.1	100.0	63.68	10273	4595.	10276	4597.

Several comments can be made about the above results, which are illustrated in Figures A.12-18 in Appendix A. The overall magnitude of fit and predict errors is much lower than for the quiet epoch due to the increased amount of data. The improvements in the predict span errors are not nearly as drastic as for the quiet epoch cases, with the exception of the medium-altitude object (25013) with a reduction in total RMS predict error from 3074 m to 350.2 m, shown in Figure A.17. Two to three-fold reductions in predict error are observed for the low-altitude and eccentric test cases in Figures A.12-15 and A.18, respectively. The high-altitude test case demonstrates the difficulty of accurately capturing drag effects at altitudes of 500 km or higher: the corrected model is actually outperformed by the uncorrected model in the predict span by 385.8 m RMS. This result, shown in Figure A.16, could indicate that the density corrections applied to higher altitudes are incorrect for this epoch, or that the DC program had difficulty converging on a good solution.

The following table presents fit statistics for the perturbed, dense data case:

Table 5.8: Fit Statistics for Perturbed Epoch Test Cases with Dense Data

NSSC #	Uncorr/ Corr	ρ_1	σ_{ρ_1}	σ_a (m)	A-priori Accuracy
09854	Uncorr	-0.086	0.605E-4	0.214	1 m
	Corr	0.024	0.883E-4	0.259	10 m
25013	Uncorr	-0.025	0.979E-3	0.087	100 m
	Corr	0.017	0.124E-2	0.089	100 m
17769	Uncorr	-0.211	0.361E-2	0.599	10 m
	Corr	0.014	0.533E-3	0.087	10 m
25947	Uncorr	-0.054	0.463E-3	0.049	10 m
	Corr	0.034	0.468E-3	0.067	10 m

5.2.5 Partial Calibration Database Test Case

This test case examined the impact of reducing the number of satellites used in the density correction database from the full 335 to 214, which is the number used by Nazarenko in his simulations [1]. The first 214 objects in order of NSSC catalog number

in the ballfcts.txt file were selected as the objects to be used to calculate the new density variations. This means that we are assuming the orbital characteristics are distributed randomly with respect to NSSC number. The following plots compare the old and new values of b_1 and b_2 over the 55-day time interval:

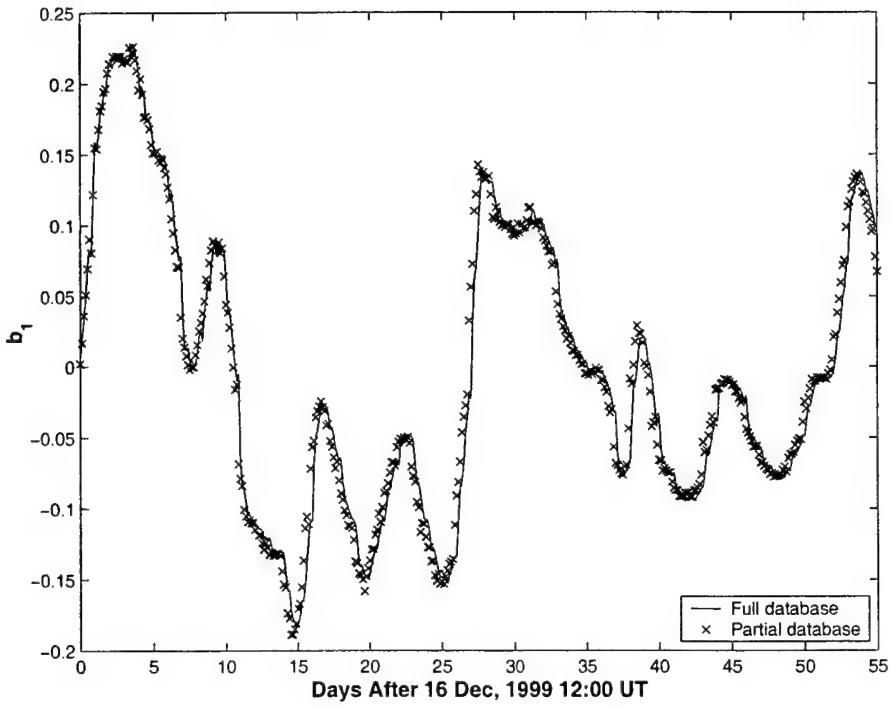


Figure 5.18: Comparison of b_1 For Full and Partial Calibration Database

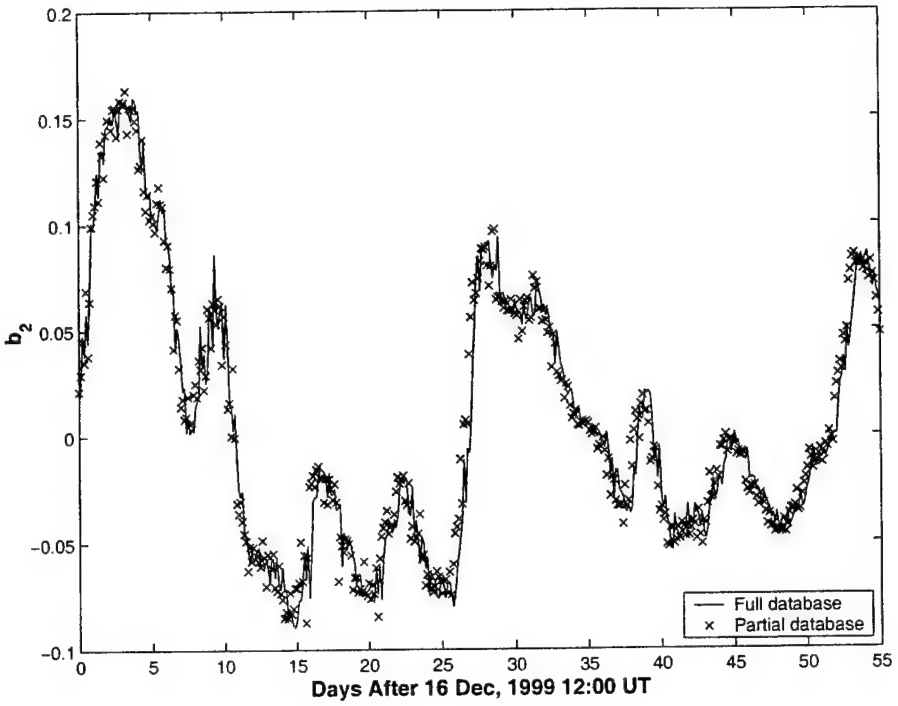


Figure 5.19: Comparison of b_2 For Full and Partial Calibration Database

If we calculate the statistics for the difference between the full and partial database versions of b_1, we find that the maximum difference is equal to 0.0775; the mean difference is 0.000113, and the standard deviation is 0.0142. For the b_2 coefficient, the maximum difference is 0.0518, the mean difference is equal to –0.000294, and the standard deviation is 0.0121. Looking at the mean values of the differences, it seems that reducing the number of calibration satellites to 214 does not seem to significantly affect the accuracy of density variations. If we consider the full database variations as the truth, the average error in the density correction factor at an altitude of 400 km is only 0.01%, increasing to 0.04% at 200 km. Even assuming that the maximum errors occur during the same 3-hour time span and in complementary directions, the correction factor will only be off by 13%. This might seem like a significant error until we realize that this worst-case correction factor is applied for only one 3-hour span.

5.3 Test of Forecasting Algorithms

The final test cases attempt to validate the density variation forecasting algorithms as presented in Section 2.3. The density variation coefficients were forecast for three-day intervals from two different epochs. The first epoch is Jan 2, 2000, which corresponds to the beginning of the predict interval for the "perturbed" sparse-data DC test cases. A comparison of the "true" and predicted coefficients is presented in Figures 5.20 and 5.21 below:

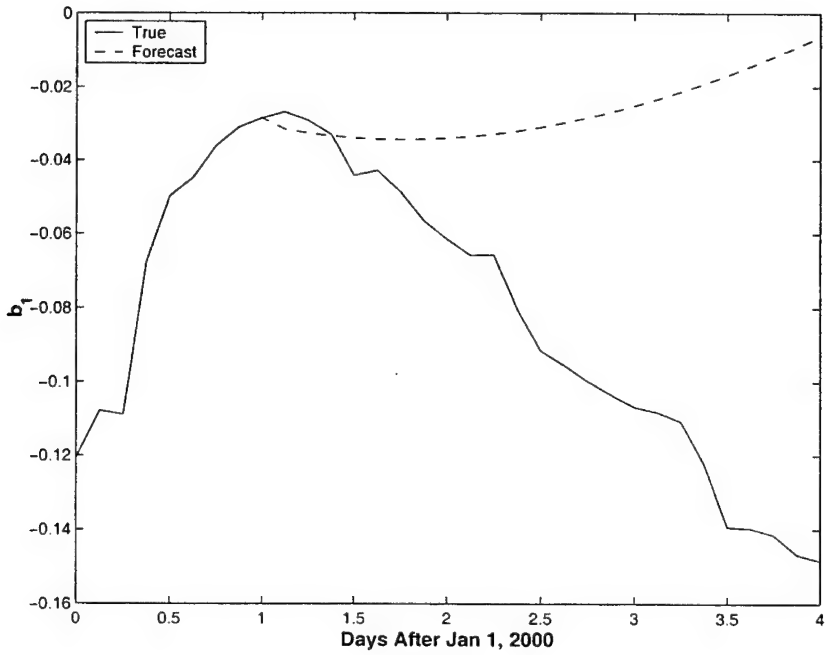


Figure 5.20: True and Predicted b_1 From First Epoch

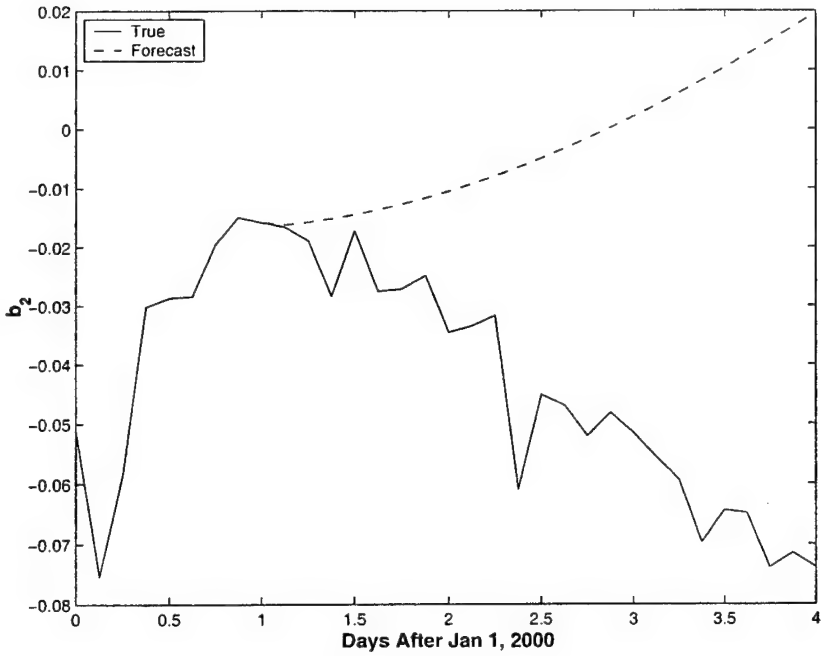


Figure 5.21: True and Predicted b_2 From First Epoch

This epoch is a particularly difficult case because the motion of the coefficients has just reversed direction, seen from the expanded time plot in Figure 5.22:

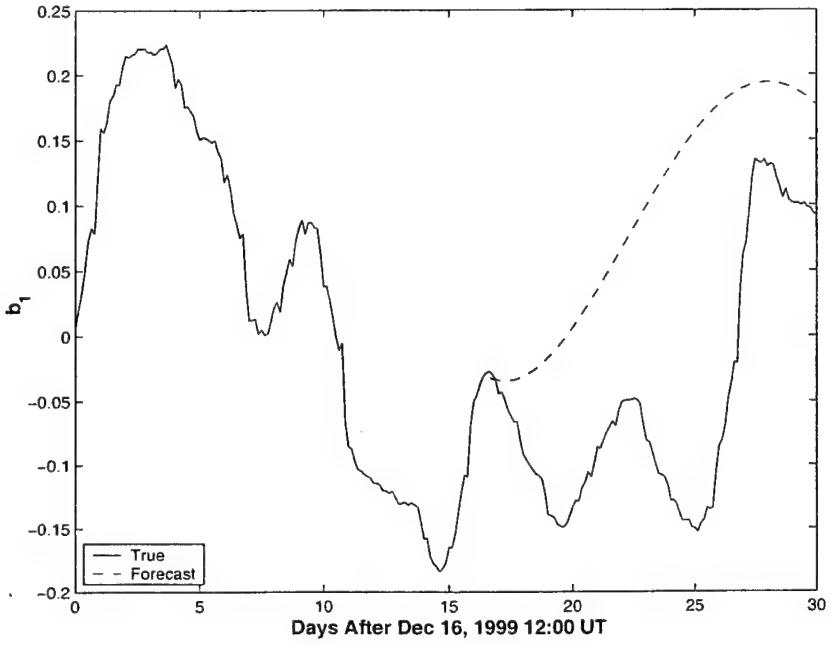


Figure 5.22: True and Predicted b_1 From First Epoch, Longer Time Interval

This is in some sense a worst case scenario because the deterministic and random components of the forecast signal are working in opposite directions.

The second epoch was arbitrarily chosen to be 25 days after the start of the overall time interval, which works out to be Jan 10, 2000 12:00 UT. Figure 5.23 illustrates the true and forecasted values of the b_1 coefficient:

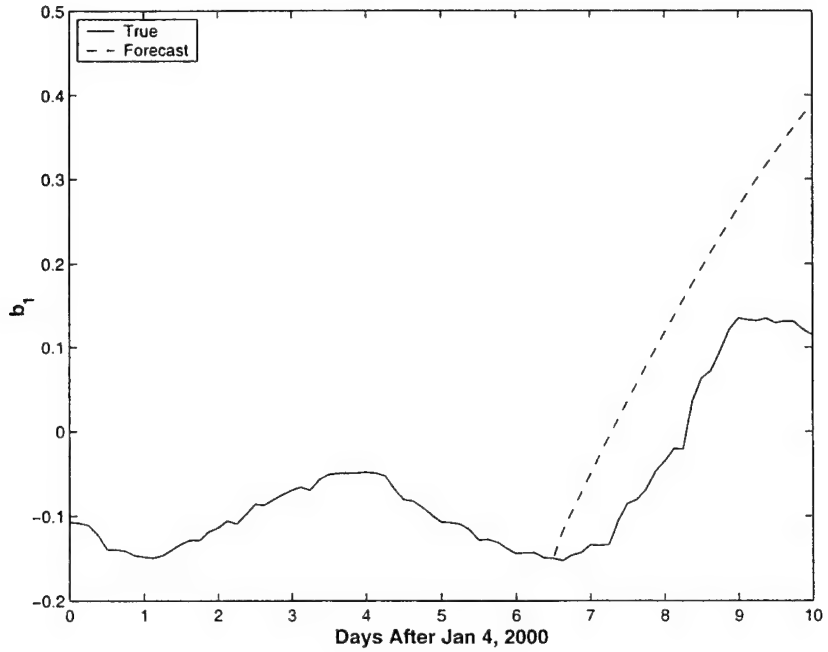


Figure 5.23: True and Predicted b_2 From First Epoch

The forecast values appear to behaving in a much more reasonable fashion for this time interval, although after about three days the forecast values begin to diverge more significantly. A conclusion we can draw from these results is that the forecasting algorithm should function reasonably well for a short predict interval (i.e. 2-3 days), but could lead to errors for longer-term predicts.

[This Page Intentionally Left Blank]

Chapter 6 Conclusions and Future Work

6.1 Conclusions

Atmospheric density modeling error accounts for much of the difficulty that space agencies have in tracking low-altitude space objects. Such diverse missions as space catalog maintenance, maneuver planning, decay and re-entry analysis, and collision avoidance could each benefit from better knowledge and prediction of atmospheric conditions. The importance of atmospheric modeling has been recognized by the scores of researchers who have worked to develop new, more accurate density models based on direct measurements of density from space or on physical principles. However, the development of new, complex models is often a lengthy and expensive process. This work has investigated a method of correcting existing atmospheric density models using information that we already possess in the space catalog database. The existing general perturbations catalog may be adequate, but the density correction process is particularly suited to special perturbations catalogs now being investigated. The method can potentially be applied to any currently existing density model and for any mission requiring better orbital prediction capability. The end result could be an "atmospheric correction service" that would allow users to obtain near real-time density corrections and predictions from a centralized location. Much work remains to be done before such a goal is attained, but this thesis has demonstrated the basic feasibility of the idea, and has laid the groundwork for future efforts.

The conclusions will be presented in three sections. The first section summarizes the analytical development of the atmospheric correction algorithms, and their implementation in a newly constructed software package. The following section will present the tools developed as part of the process, and the modifications and improvements made to the GTDS software utility. The final section of conclusions will outline the numerical analysis and testing of the density correction process. A section will follow the conclusions on proposed future work.

6.1.1 Algorithm Development and Implementation

The first step in atmospheric correction research was the detailed derivation and analysis of the basic algorithms originally presented in the work of Nazarenko [1]. No major errors were found in any stage of the process. Notation was standardized, and some of the equations were simplified or explained in more familiar terminology. A few errors were removed, primarily in the algorithms dealing with the forecasting of the density variation models in Section 2.3.

The algorithms were implemented and tested in an entirely independent software environment. The Matlab software package [57] proved to be invaluable for its ability to manipulate large amounts of data in matrix form. The new implementation served as an excellent validation of Nazarenko's investigations and of the feasibility and robustness of the algorithms.

Another important conclusion deals with the interaction among researchers in different locations around the country, and indeed, the world. Using modern communications technology such as e-mail and high-capacity portable storage mediums, we were able to exchange data, ideas and suggestions as the research progressed. This work would not have been possible if many very knowledgeable scientists and engineers had not been willing to freely share information and insight.

6.1.2 Tools and Software

A very useful result of the research in this thesis was the establishment of a powerful, flexible computational environment for orbit determination and analysis on the DC1 Unix-based SGI workstation at The Draper Laboratory. The most current functional version of R&D GTDS on the personal computer was ported to DC1 and thoroughly tested for all necessary functionality. The source code was imported into a configuration-managed environment so that future modifications to GTDS will be fully documented and reversible. It should be noted that GTDS has existed in configuration managed environments up until the last few years, when a proliferation of different versions came into existence. It is highly desirable for all versions to be re-imported or merged into one fully functional platform. This work can serve as the first step in such an effort.

A complete reference has been compiled with instructions on how to use the configuration management system, build GTDS data files, link and compile the source code into an executable, and operate the software.

Due to the vast amount of computation and data handling required by the atmospheric correction process, it was necessary to develop automated, efficient techniques to drive GTDS runs and to generate and parse large data files. The Perl programming language was chosen for this purpose and quickly proved itself to be indispensable. Perl is portable and widely used, meaning that the scripts given in Appendix C may be used on different platforms and easily tailored to specific computational environments. A benefit of the use of Perl on DC1 was the relative ease of implementing large jobs in parallel, thereby reducing run time by an order of magnitude. It became possible to execute thousands of high-precision differential corrections for hundreds of objects in the database over an interval of several weeks or months and finish in a matter of hours.

Implicit in the automated Perl scripts is the capability of using GTDS in a more flexible manner. The script files can generate GTDS card decks, run the GTDS code, and extract data from the output files as necessary. The TLE2osc.pl program produces osculating orbital elements from an input file consisting of Two-Line Elements sets. The genobs.pl program has the capability of producing simulated observations in OBSCARD format with user-defined station locations, biases, and observation noise. The estbfs.pl program can perform iterative differential correction runs for large numbers of objects and over long intervals of time.

GTDS was modified and improved to make the code more functional and reliable. The ability to generate state histories in ASCII file format, present in the VAX-GTDS version, was added to UNIX-GTDS. Bugs associated with the year-rollover problem and the no-observation problem as described in Section 3.1.4 were corrected. The ability to run GTDS in parallel, with simultaneous access of data files, was implemented and verified. Finally, a list of known bugs and functional limitations of the code was compiled and is presented in Appendix B Section 5.

6.1.3 Density Correction Analysis

The ability of the density correction process to update a density model in near-real time was demonstrated over a range of altitudes and atmospheric conditions. The algorithms were shown to be capable of removing time-localized errors in spans of one day or longer. When actual values of uncorrected and corrected density for both geomagnetically quiet and perturbed epochs were compared, it was found that biases in density values were reduced to 2% or less and standard deviations cut by 21-36%.

The reduction in density error translated to significant improvement in fitting observation data and reduction in orbit prediction error. Four types of orbits at different altitudes and eccentricities were investigated in both quiet and perturbed epochs and with low and high data rates. Density correction led to better fits of sparse data, and in some cases allowed the differential corrections process to converge where the use of uncorrected density was causing divergence. The onset of quadratic error growth in the predict span was delayed for one-two days, and total error after three days was often reduced by an order of magnitude or better. Drag coefficients could be more accurately estimated from observation data, which indicates potential applications in estimation of unknown spacecraft characteristics such as mass or cross-sectional area.

Algorithms for forecasting density correction models were tested for quiet and perturbed epochs. The methodology of separating the signal into random and deterministic components was validated, and reasonably good forecasting was demonstrated for intervals of up to three days. The architecture for the processing of actual observations was established. Overall, the techniques presented in this thesis exhibited flexibility and a degree of robustness for different conditions, altitudes, and types of orbits, and show great promise for future investigations.

6.2 Future Work

All future work to follow is ordered first by topic and next by priority within the topic.

6.2.1 Algorithm Improvements

Most of the work on implementation of density correction algorithms has been performed, with the exception of estimation of "true" ballistic factors and the improvement of the forecasting algorithms using more sophisticated physical models. The former, as outlined in Section 2.2, was originally intended for inclusion in this work, but was not accomplished due to time constraints.

Estimation of "true" ballistic factors is the last major step in the simulation tests. If the density correction process is shown to be effective even when given inaccurate ballistic factors for the majority of the objects in the database, the simulation tests will be essentially complete, and we can move on to processing of actual observations.

The forecasting algorithms as derived in Section 2.3 are reasonably effective, but do not incorporate very much information about the physical processes involved in the evolution of atmospheric conditions. Very detailed methods of forecasting solar and geomagnetic indices up to a month in advance have been presented in a study commissioned by the European Space Agency in 1991 [58]. These techniques may be incorporated into the density variation forecasting algorithms, or possibly can be implemented as a separate tool used to generate predicted solar and geomagnetic indices for direct input into an atmospheric model such as JR-71.

6.2.2 Software Additions

When examining actual values of density produced by the JR-71 model in Section 5.2.2, it was noticed that the density moved in a discontinuous fashion for each three-hour value of ap. After some investigation, it was determined that these jumps were due to empirical equations presented on page 37 of Jacchia's 1971 report [15]. Jacchia recommended that smoothed geomagnetic indices are used in the equations, but the implementation of his model in GTDS was found to use the original discontinuous values. For the simulated data cases that were investigated in this thesis, both the "truth" and smoothed models contained these jumps, although frequently in different directions and magnitudes. However, if actual space catalog data is to be analyzed, it will be necessary to implement a smoothing process in GTDS in order to avoid errors in orbital

estimation. Other organizations using some form of Jacchia's 1971 model should ensure that this error is not present in their orbit propagation software as well.

The next step in improvement of software should be to add a sequential filtering capability to the density correction software. This will require numerous modifications to the estbfs.pl Perl script, and may require some validation of the filter code in GTDS. A sequential filter, however, will eliminate the need of processing the same data multiple times in overlapping batch runs, and will allow more up-to-date estimates of ballistic factors for each object in the database.

The multiprocessing capability of the Perl scripts relies largely on the inherently parallel nature of the DC1 machine. If the density correction algorithms are to be implemented in a truly portable, non-platform dependent fashion, the use of a parallel processing standard such as MPI (Message-Passing Interface) [59] or PVM (Parallel Virtual Machine) [60] will be required.

Most of the research done by Nazarenko and his colleagues used the GOST atmospheric model, an empirical density model first developed by I.I. Volkov in the late 1970s and refined in 1984 [47]. It would be very useful to implement this model into GTDS so that some of our results could be compared more directly to earlier density correction investigations. The relative simplicity and accuracy of this model also makes it desirable for other applications.

The final area of software improvement deals with GTDS. The first necessary step should be to investigate the known bugs and limitations of the code presented in Section B.4, and to determine if they are causing any degradation of results. Once these bugs and limitations are removed or bypassed, future work should focus on the incorporation of the additions currently existing in the GTDS PR-6 version on the PC. Many of these additions are desirable for work on atmospheric correction, including atmospheric lift modeling, the altitude-dependent error function, and filter improvements.

Other modifications to GTDS exist only in the version currently residing on the VAX, including the J2000 coordinate system, 50x50 gravitational models, and solid Earth tides. These modifications, if incorporated into UNIX-GTDS, will aid in the processing of actual observational data.

Nazarenko, Yurasov, and others are currently investigating the Universal Semianalytic Method (USM) semianalytic theory. It would be desirable to incorporate this software as part of GTDS or as a standalone software package. This would encourage further cooperation with their efforts, and would provide Draper Laboratory with another powerful, highly accurate tool for orbit determination and prediction.

6.2.3 Further Tests of Density Correction

The tests presented in this thesis have demonstrated the effectiveness of density correction under various conditions and for various types of objects, but more testing is required. A number of trade studies should be performed using simulated data, including:

- Data rates: can higher observation rates for the objects in the density correction database help capture short-term (i.e. 3-hour) density variations?
- Perturbation models: general perturbations vs. special perturbations
- Gravity models: what is the effect of gravity model truncation?
- Correction of different density models: MSISE-90, GOST
- Biases/measurement error: how do observation biases and noise affect the results?

Once these tests have been performed, the process can be applied to problems involving actual data. This will inevitably require extensive analysis of the error and bias characteristics of the data, but the overall data flow should follow the outline presented in Section 4.2. The software has been designed to allow for actual data processing with minimal modification.

The processing of actual data may require the use of a sequential filter. If so, the filter should be validated using simulated data before its application to real-world problems. As mentioned in Section 6.2.2 above, PR-6 GTDS contains a number of fixes and improvements to the Linear Kalman Filter (LKF) and Extended Kalman Filter (EKF). These additions should be a starting point for any filtering analysis.

Once the above validations and additions have been made, the concept of the "density correction service" can be investigated in detail. Each phase of the atmospheric

correction process will be implemented in a more operational environment. There are several requirements for operational near-real time atmospheric correction, including:

- Obtaining up-to-date observation data
- Keeping a current database of object characteristics, such as mass, cross-sectional area, true ballistic factor, and status as standard or non-standard
- Obtaining current and predicted solar and geomagnetic indices

These requirements are challenges in themselves, but are necessary if a true test of concept is desired. An operational test could feature a number of targets with unknown mass or shape characteristics, and the objective would be to use density correction in near-real time to improve orbital prediction or identification of these objects. The test could be run over a period of several months, with the atmospheric correction service providing real time or forecast corrections to the user on demand. The density corrections could even be made available on the Internet, as are many orbit determination products such as TLEs or solar/geomagnetic indices. If successful, the atmospheric correction service could be expanded to include multiple atmospheric models. The service could be used by organizations worldwide to improve tracking, maneuver planning, collision avoidance, or for whatever other purposes that are desired.

6.2.4 Application of Density Correction to New Problems

If the density correction process can be validated as a truly effective and robust technique for removing errors in density models, the number of potential applications is vast. One application which has recently taken on more importance is collision avoidance and debris analysis. With the ever-expanding number of objects in Earth orbit, the risk is growing that a space asset such as the soon-to-be inhabited International Space Station (ISS) will be severely damaged or destroyed by collision with debris. It is therefore very important that we obtain the capability of predicting the orbits of low-altitude debris with more precision. Improved or corrected atmospheric models will also allow us to execute more effective avoidance or stationkeeping maneuvers with less use of propellant.

Part of the problem of debris avoidance is the identification and estimation of characteristics for newly-acquired space objects. A more accurate atmospheric density

model will allow for better estimation of mass, area, or coefficient of drag, which in turn improves prediction capability.

The goal of this research was to provide a method of obtaining better atmospheric density estimates without building a new model. However, if a large amount of correction data specific to a particular model can be compiled, it may be possible to systematically remove some biases from the model equations.

The past thirty years have seen great advances in our understanding of the atmosphere, but not in our ability to model it. However, with new high-speed computers and more accurate orbit propagation and estimation techniques, we are finally beginning to make effective use of the vast amounts of observational data collected every day. The only challenge now is to ensure that information is distributed freely and ideas allowed to circulate throughout the space surveillance community. If we can meet that challenge, it seems inevitable that the "15% barrier" in density model accuracy can finally be broken.

[This Page Intentionally Left Blank]

Appendix A Data Analysis Plots

This appendix contains additional plots of test case results that are presented in Chapter 5.

A.1 Schatten Mismodeling Quiet Epoch Test Cases

Figure A.1: NSSC# 09854 Fit Span Error, No Corrections

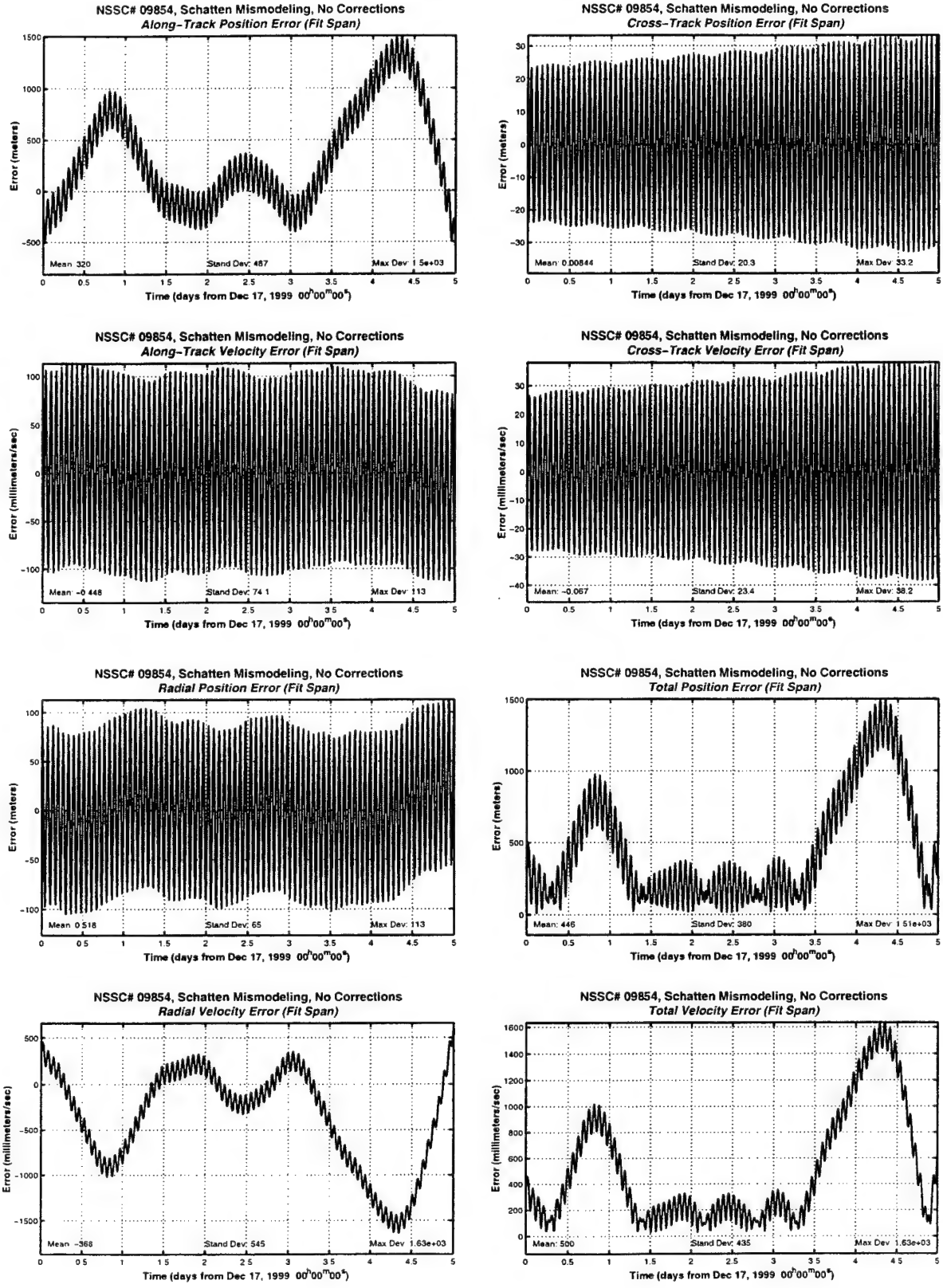
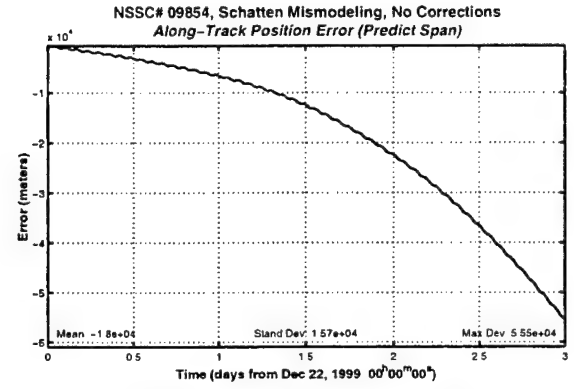
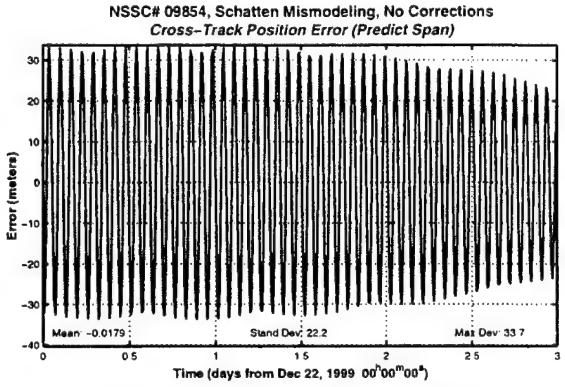
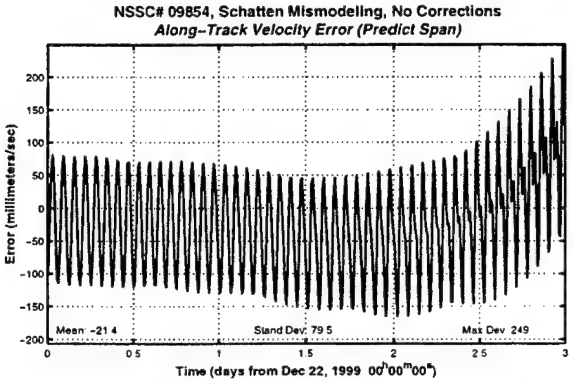
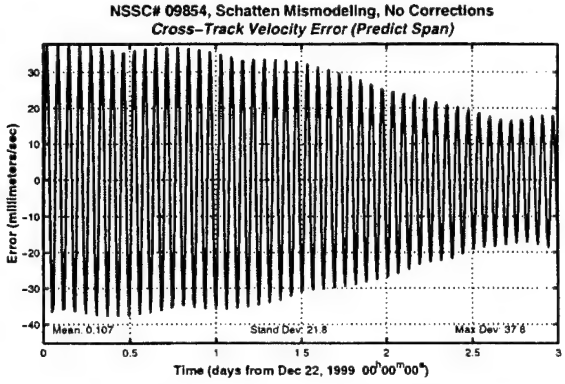


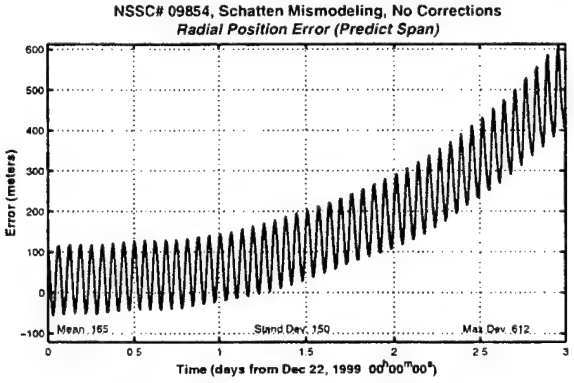
Figure A.2: NSSC# 09854 Predict Span Error, No Corrections

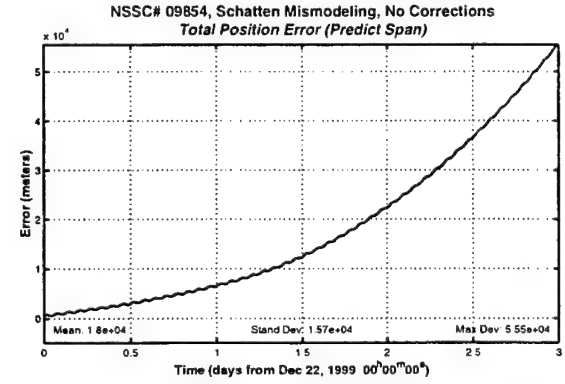


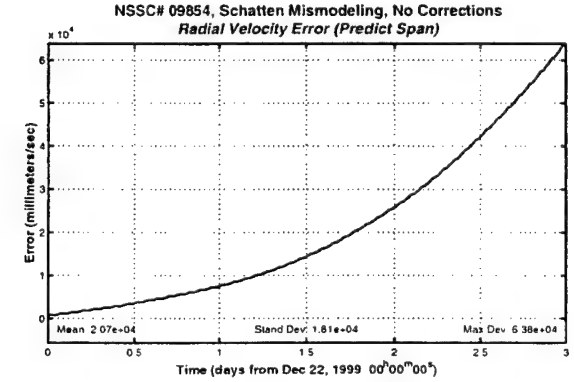












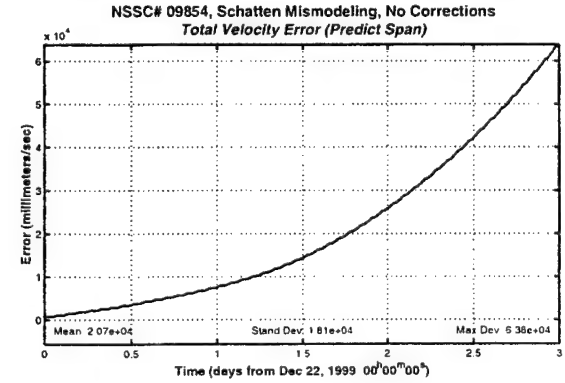


Figure A.3: NSSC# 09854 Fit Span Error, With Corrections

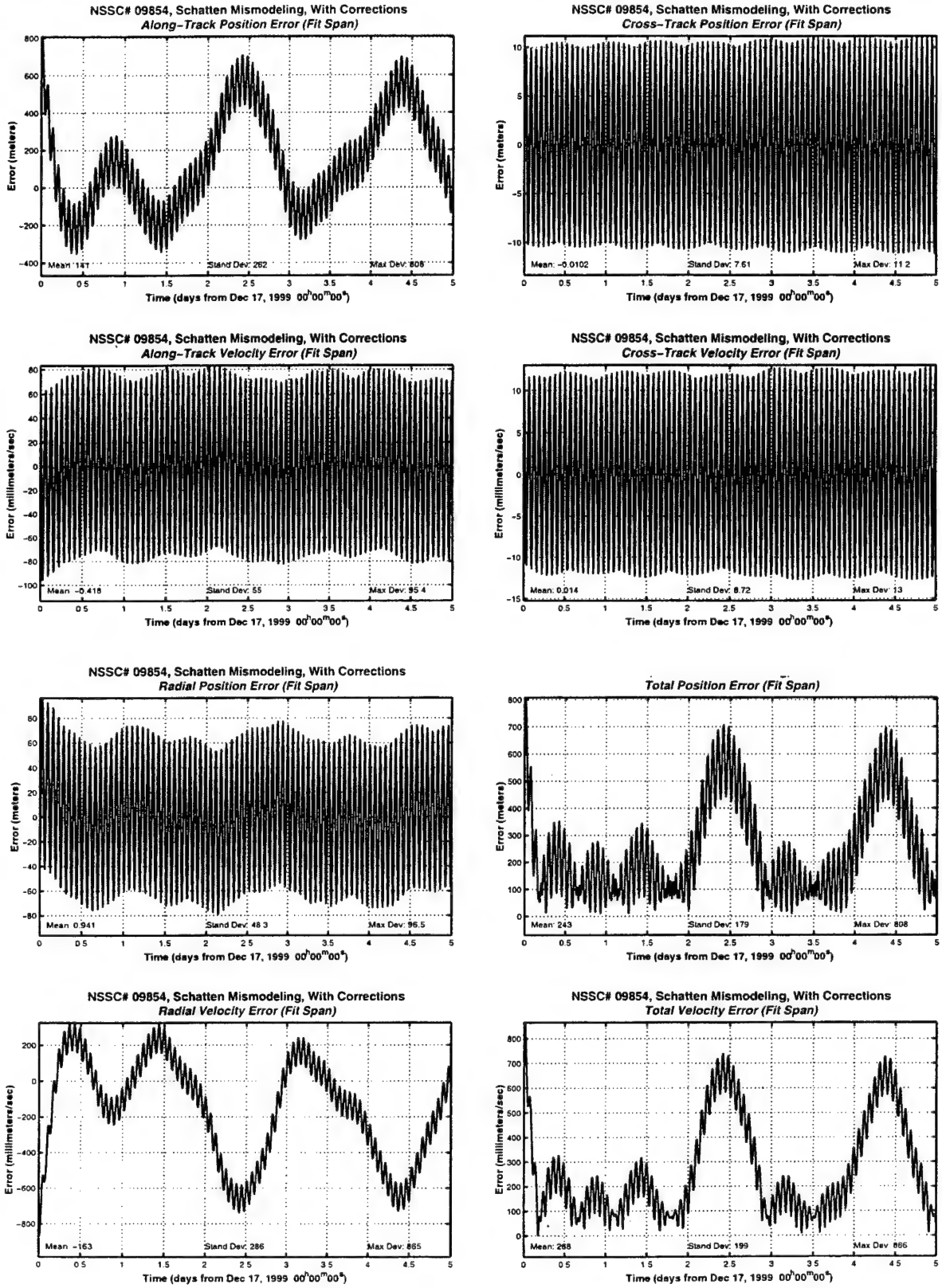
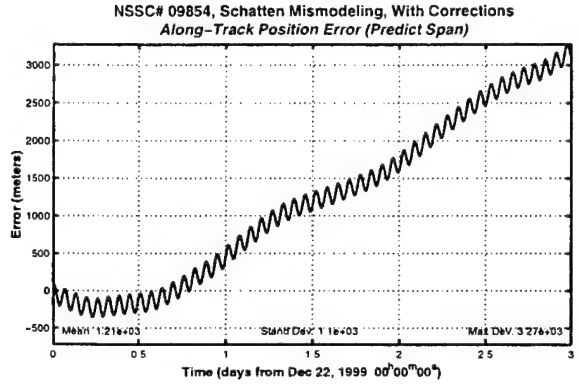
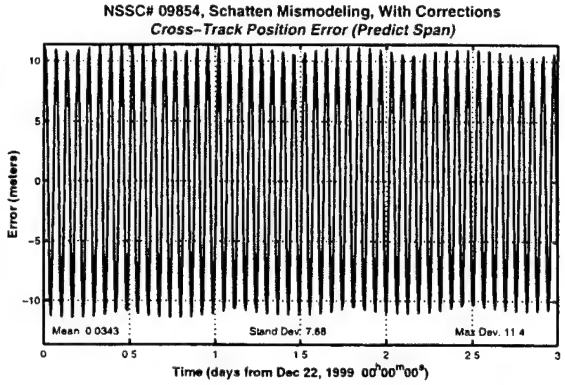
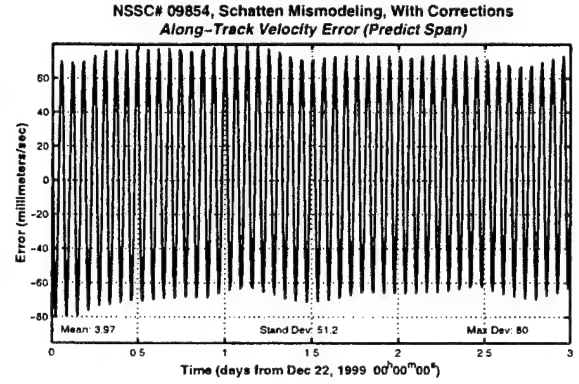
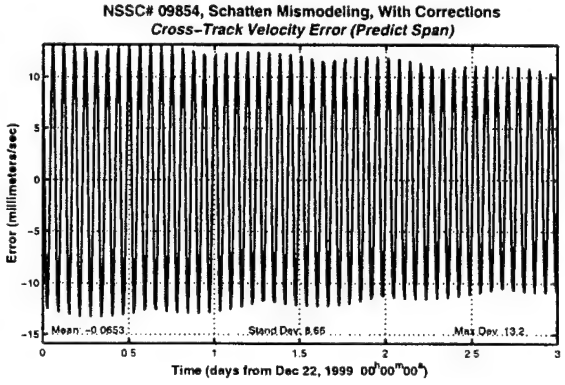


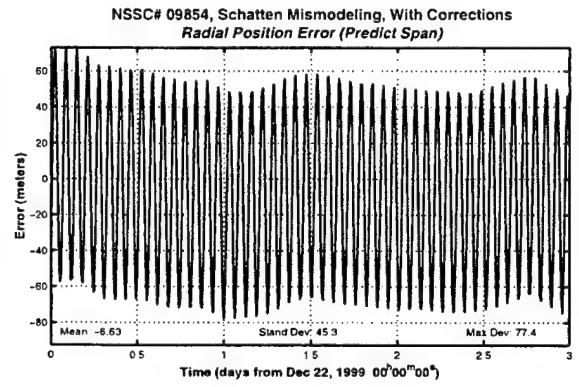
Figure A.4: NSSC# 09854 Predict Span Error, With Corrections

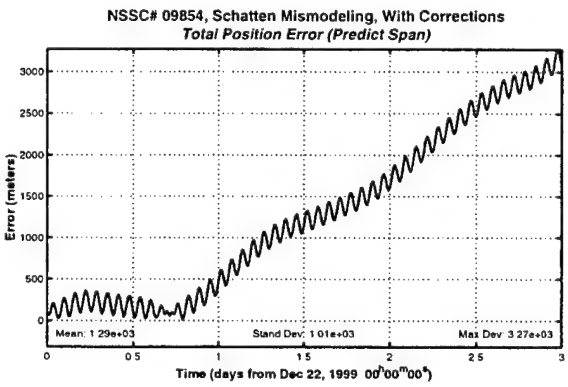


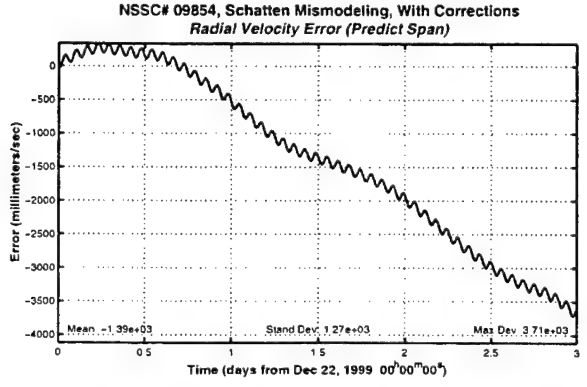












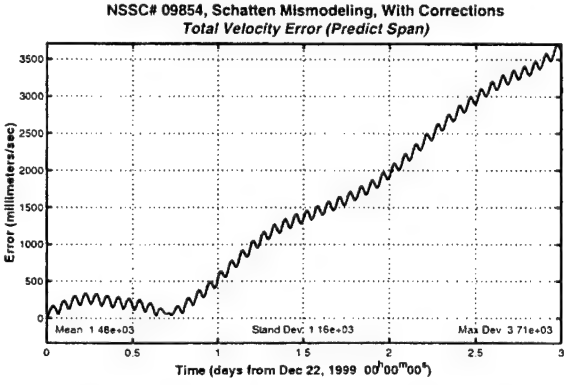
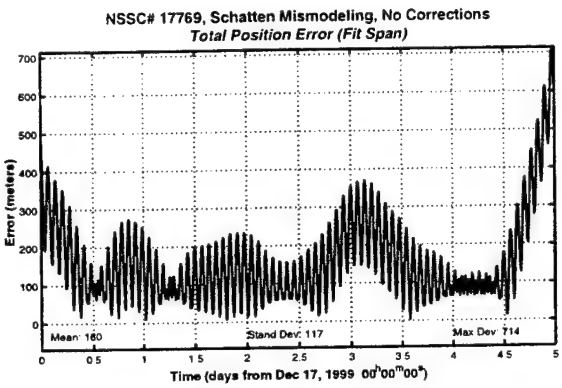
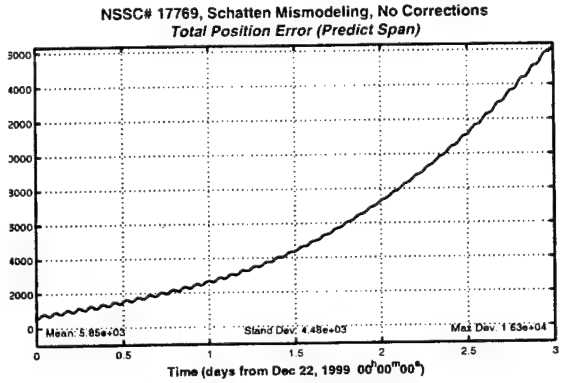
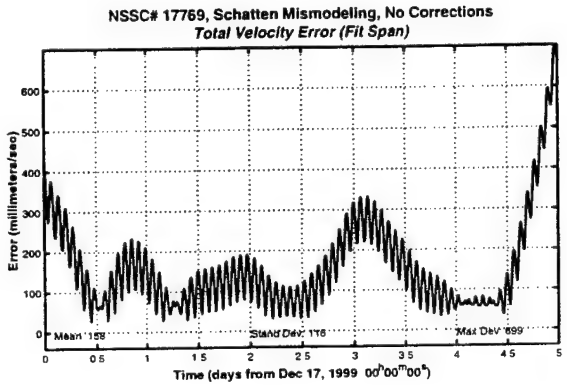
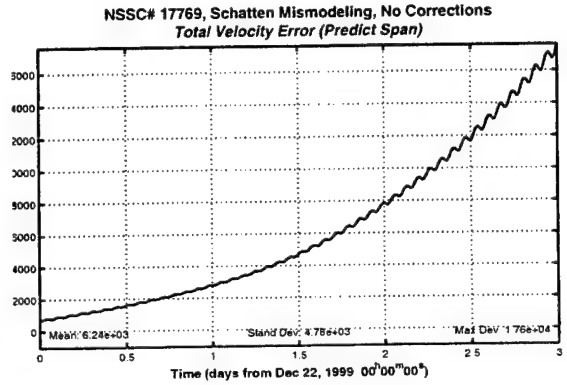


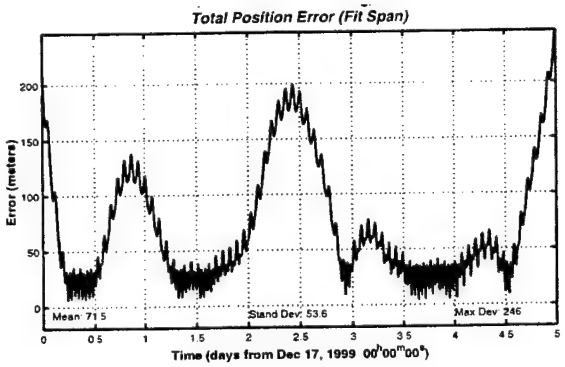
Figure A.5: NSSC# 17769 Fit and Predict Span Error, Without/With Corrections

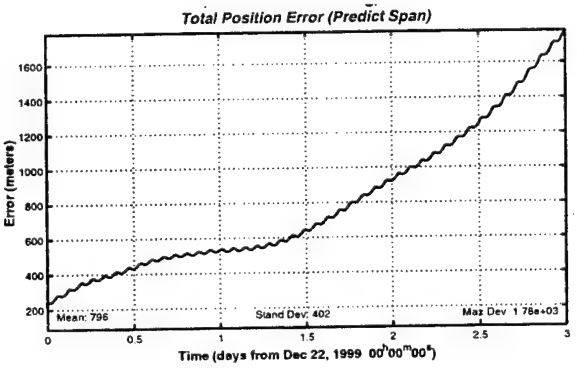


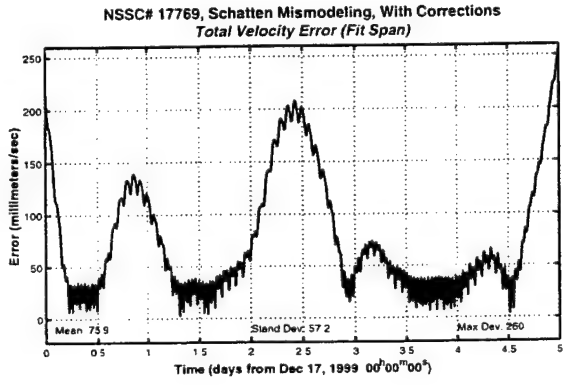












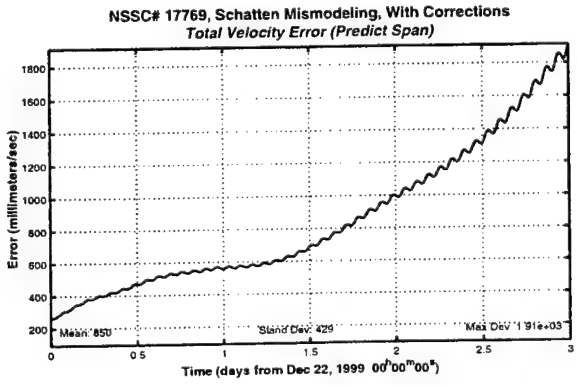
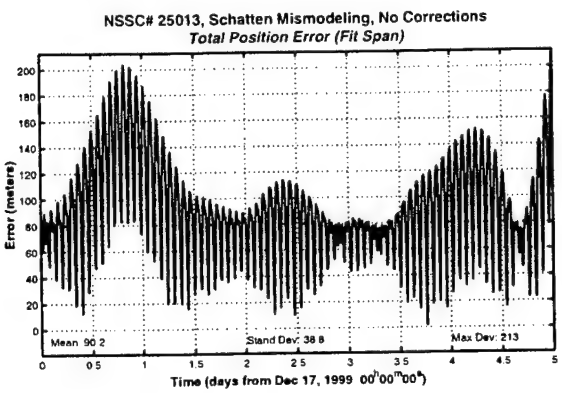
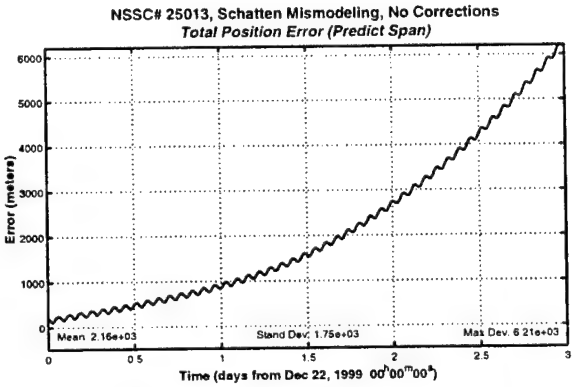
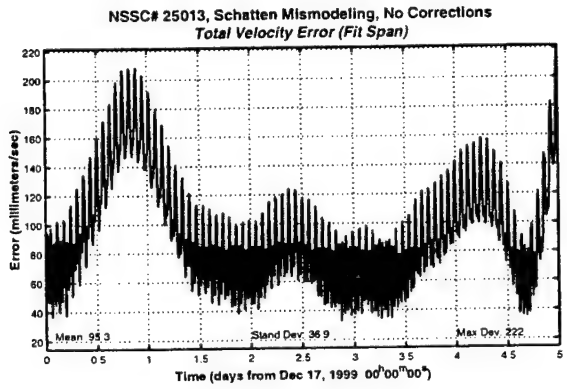
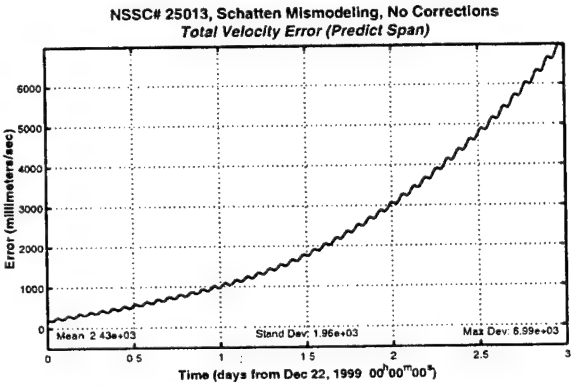


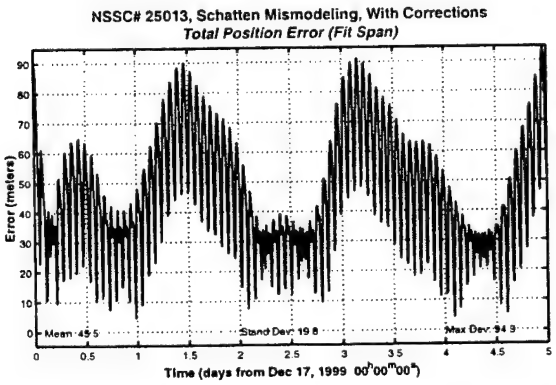
Figure A.6: NSSC# 25013 Fit and Predict Span Error, Without/With Corrections

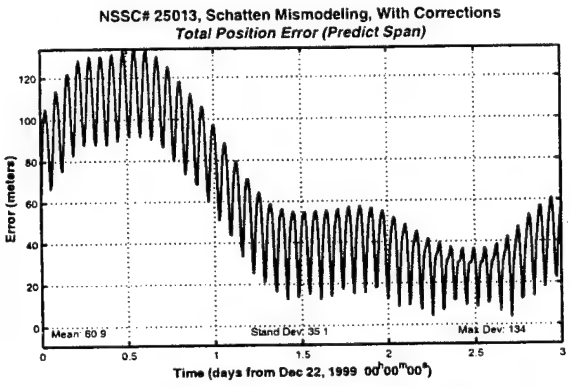


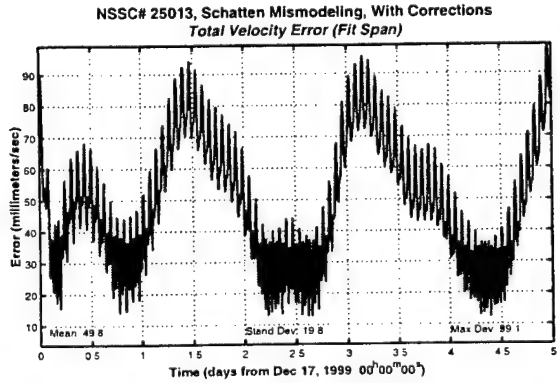












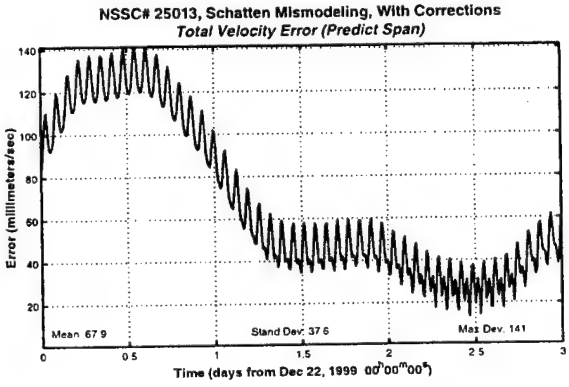
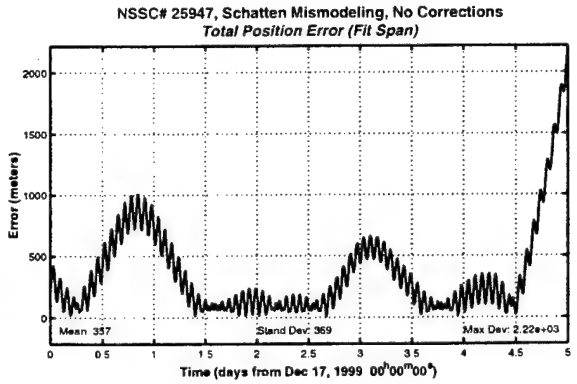
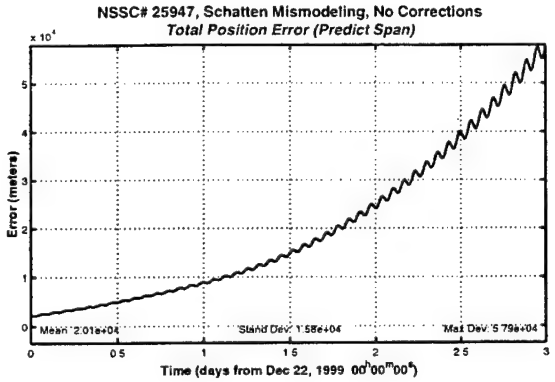
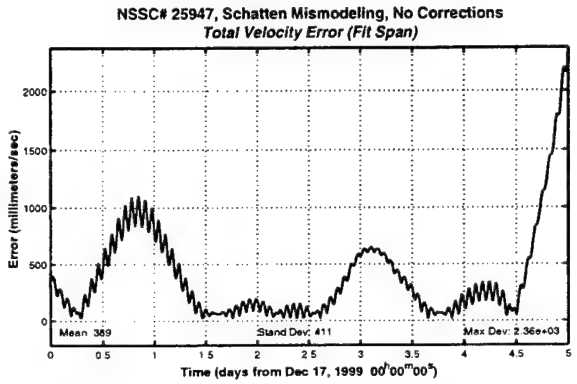
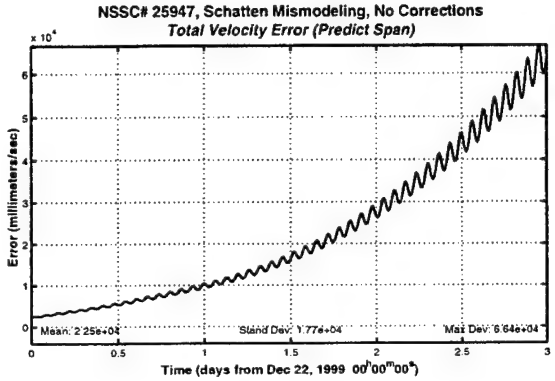


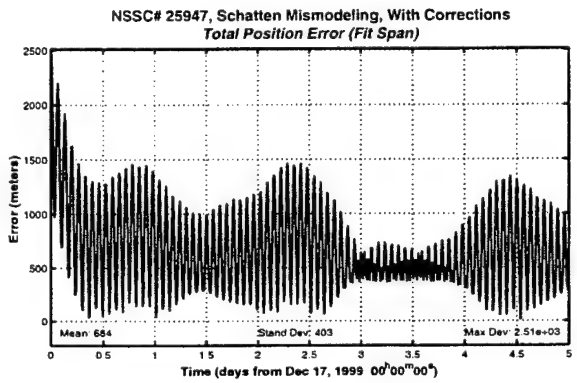
Figure A.7: NSSC# 25947 Fit and Predict Span Error, Without/With Corrections

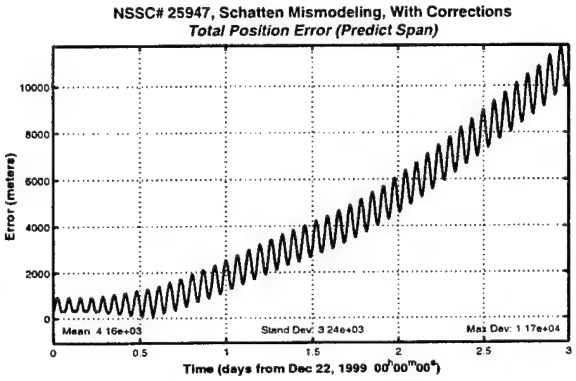


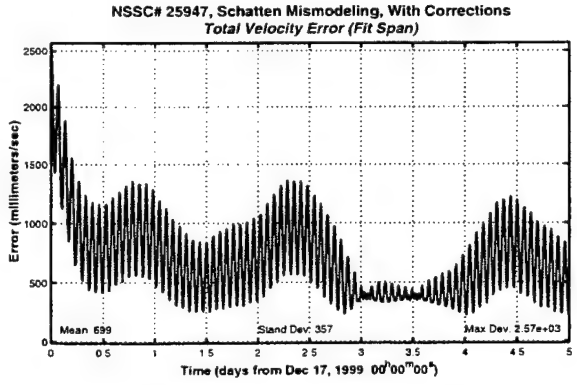


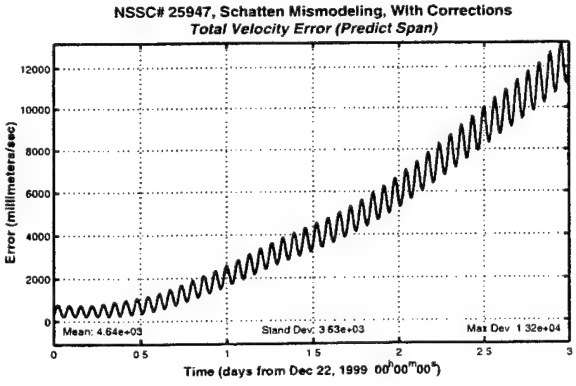












Schatten Mismodeling Perturbed Epoch Sparse Data Test Cases

Figure A.8: NSSC# 09854 Fit Span Error, With Corrections

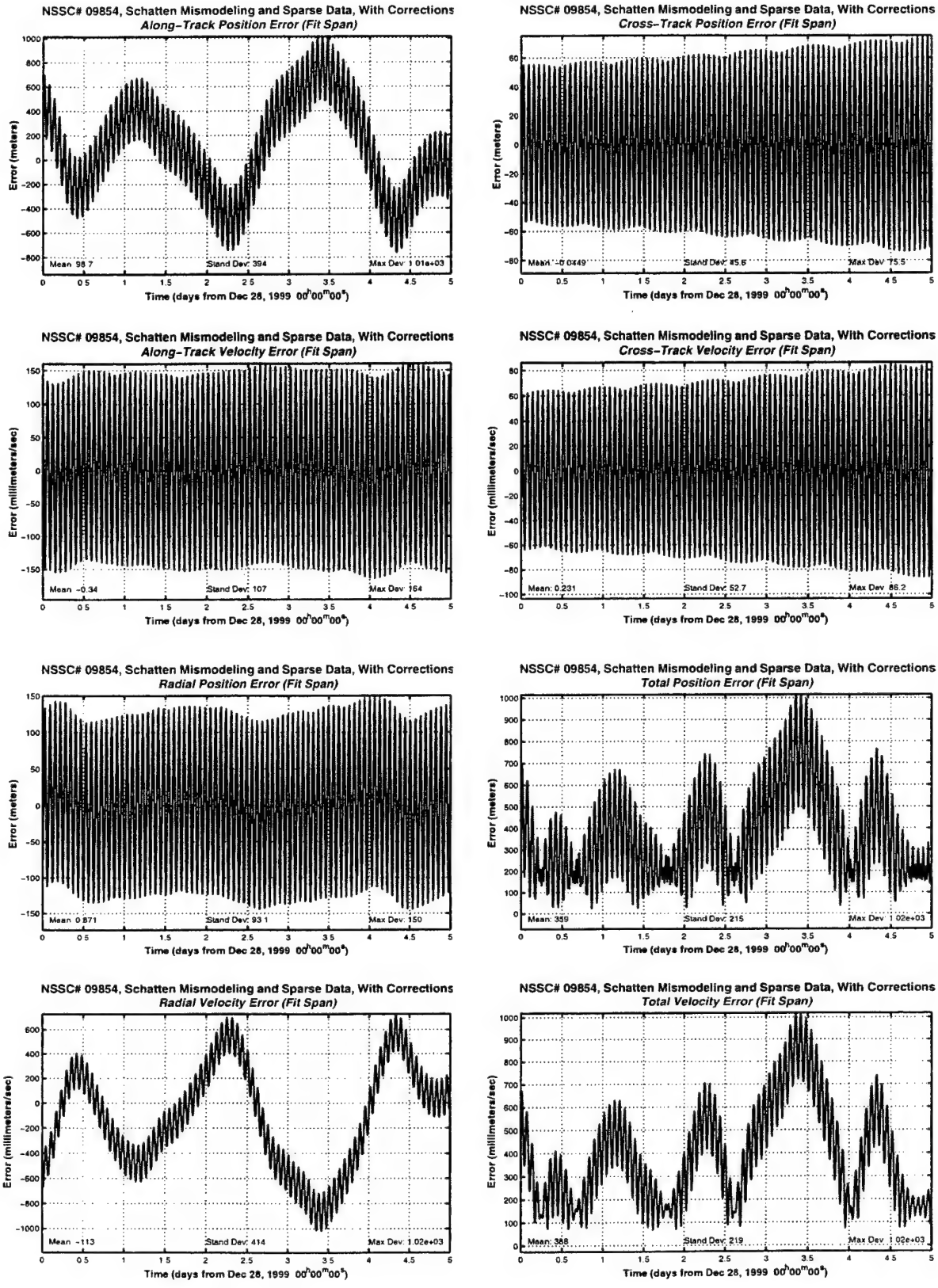


Figure A.9: NSSC# 09854 Predict Span Error, With Corrections

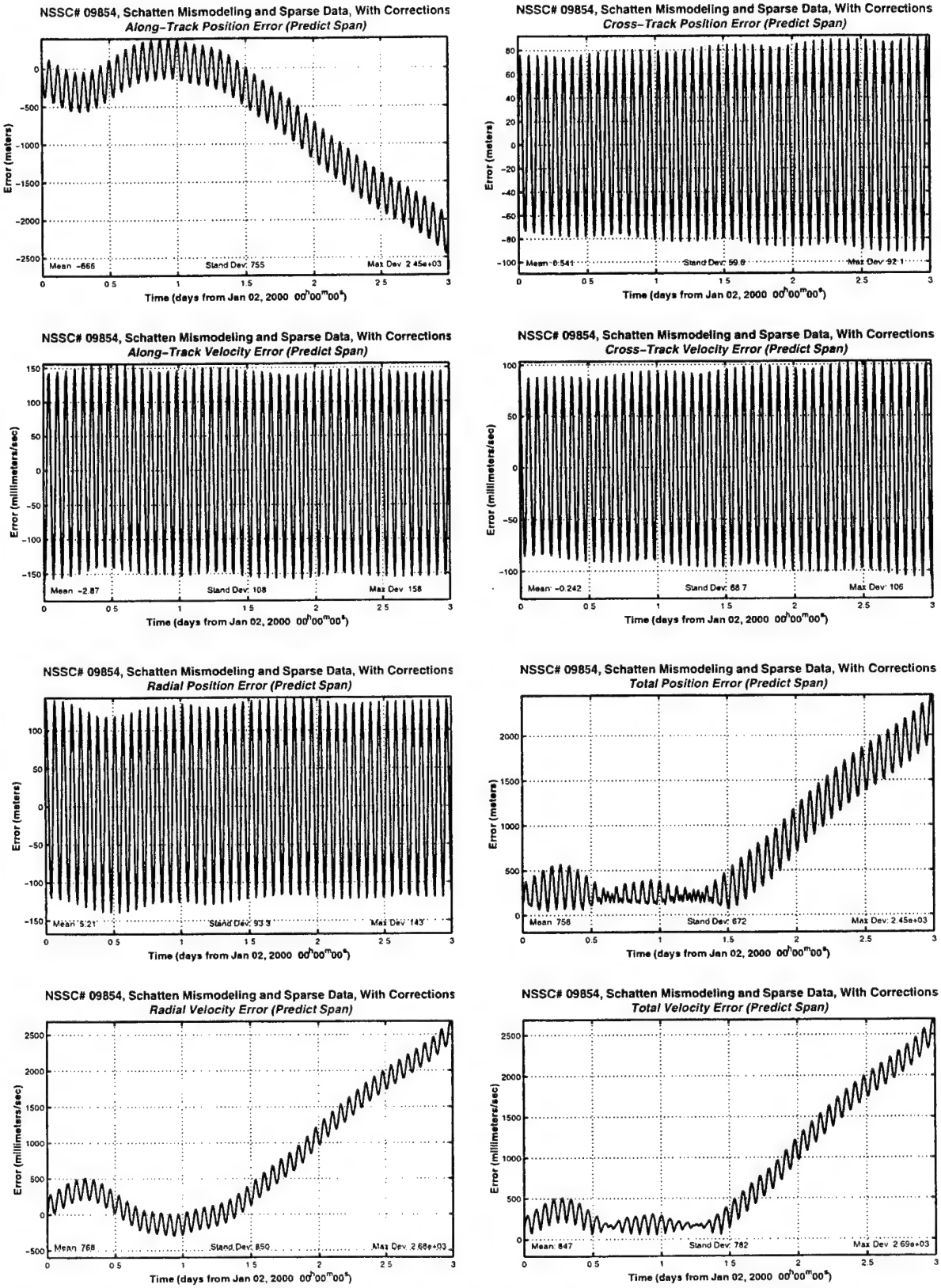
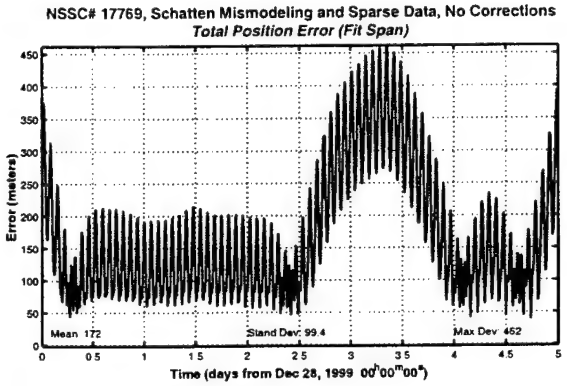
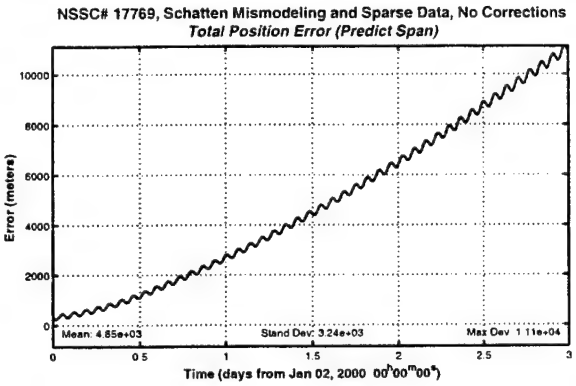
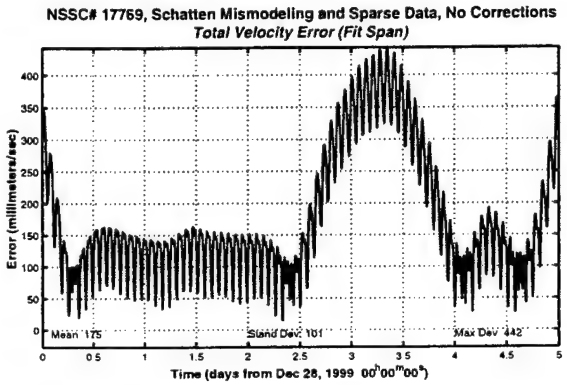
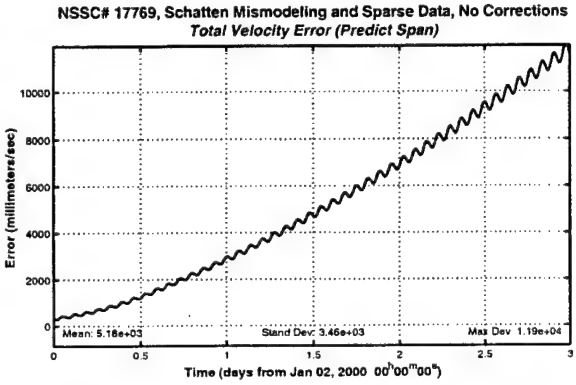


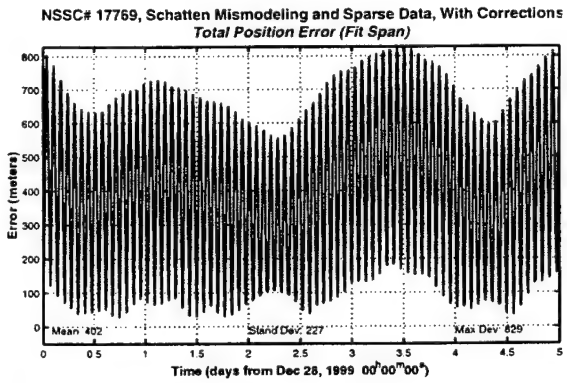
Figure A.10: NSSC# 17769 Fit and Predict Span Error, Without/With Corrections

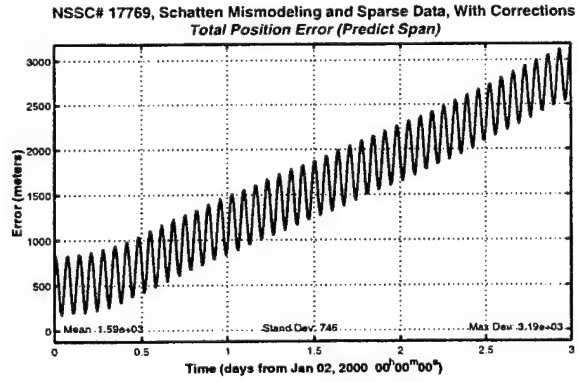


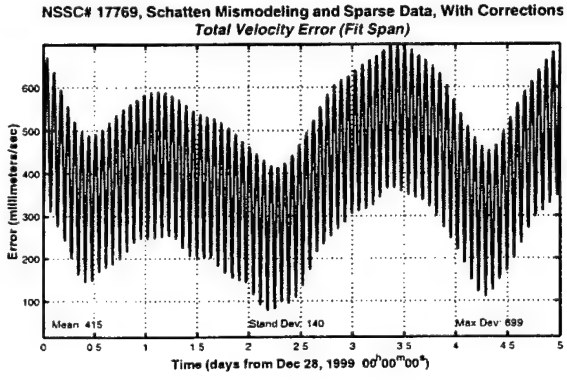












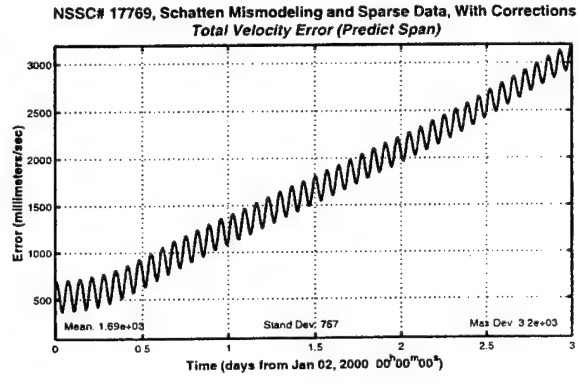
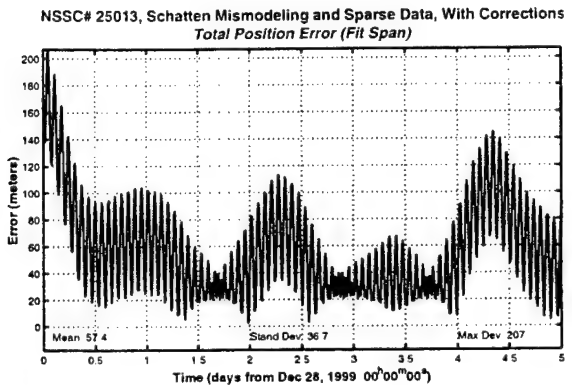
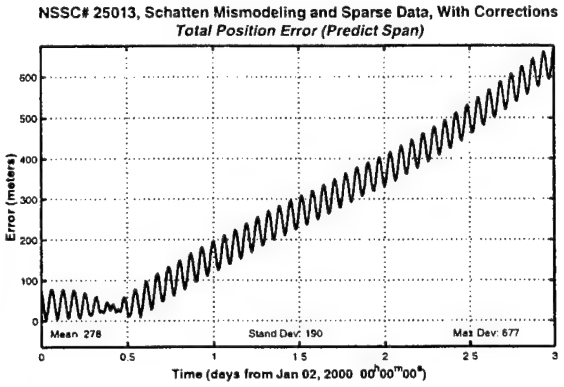
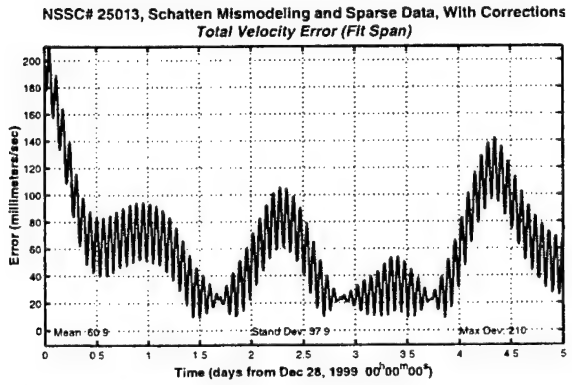
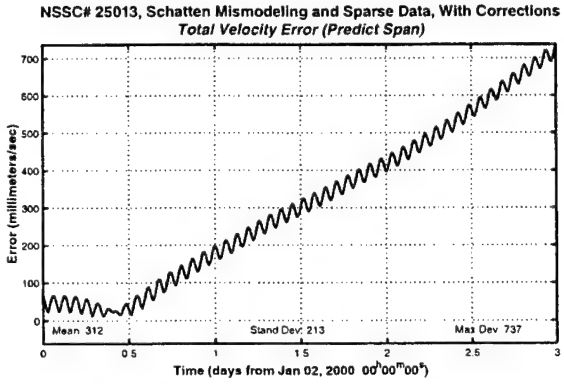


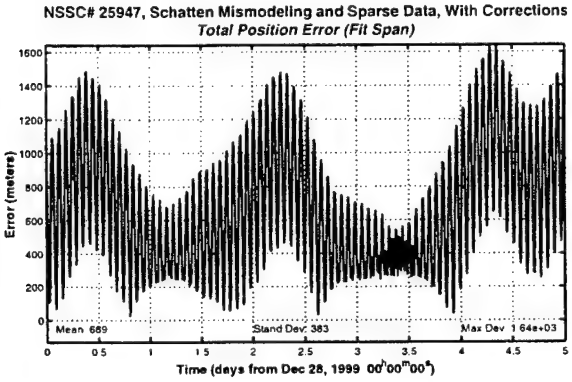
Figure A.11: NSSC# 25013 & 25074 Fit and Predict Span Error, With Corrections

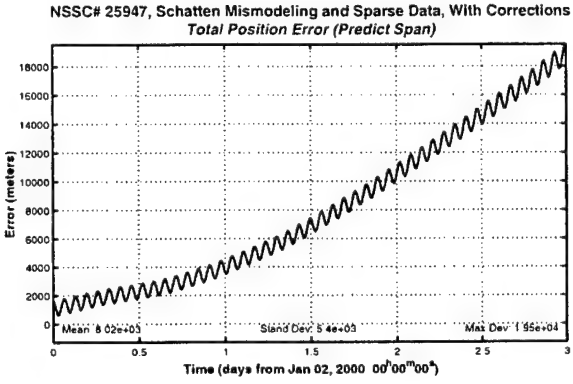


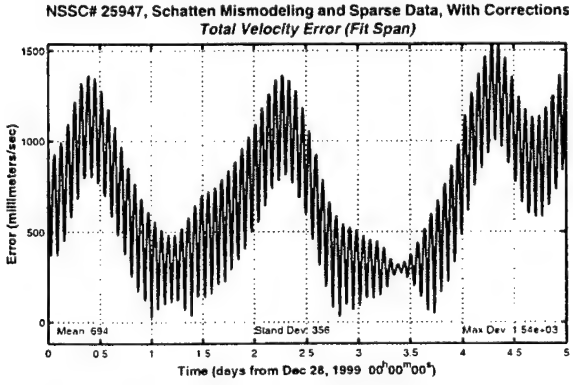


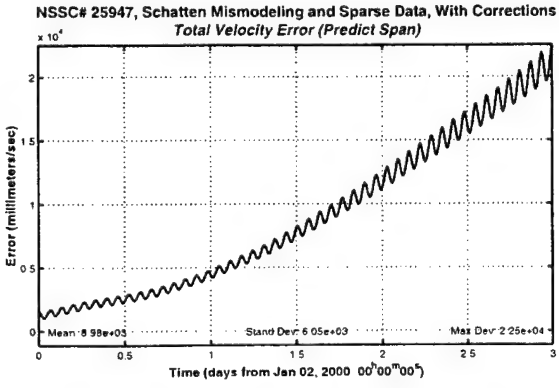






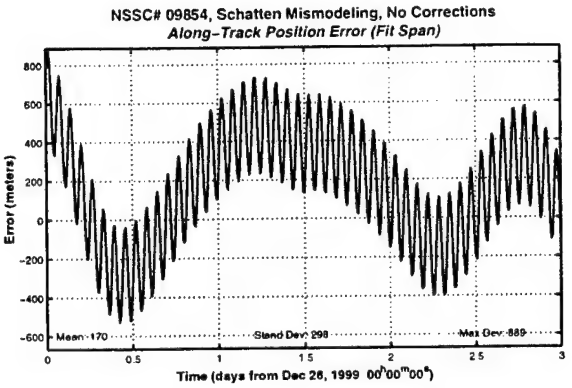


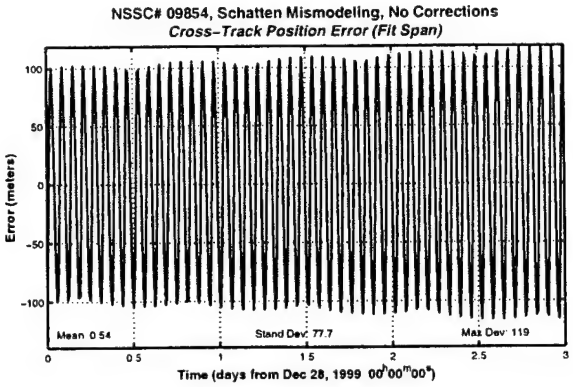


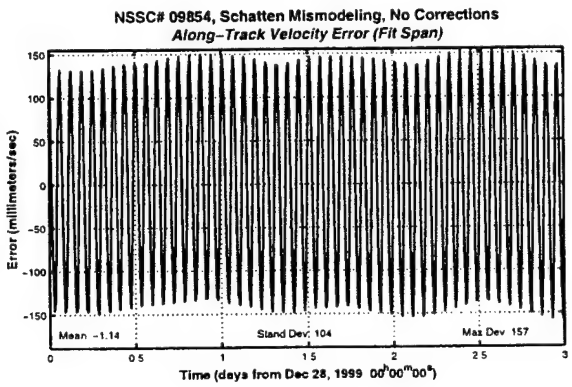


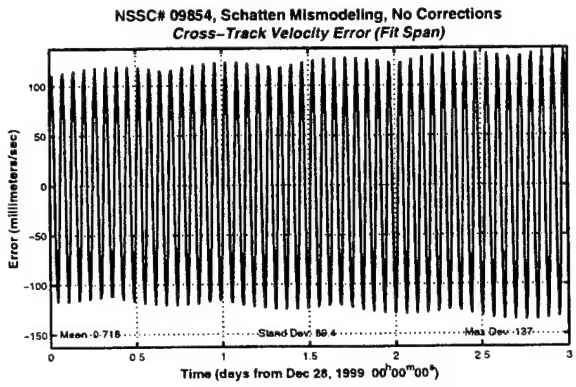
A.3 Schatten Mismodeling Perturbed Epoch Dense Data Test Cases

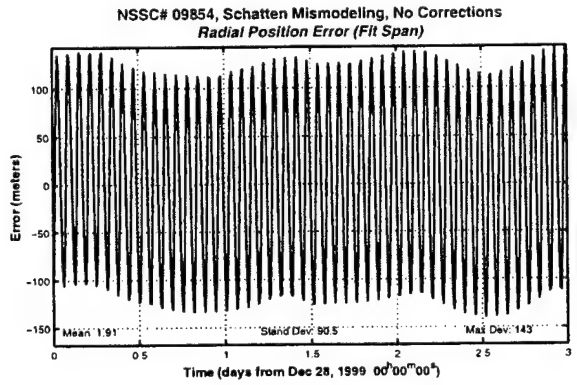
Figure A.12: NSSC# 09854 Fit Span Error, No Corrections

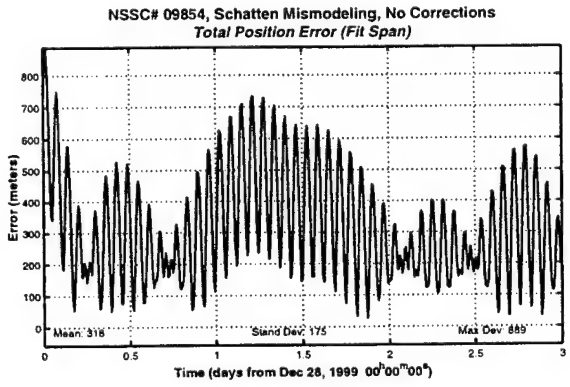


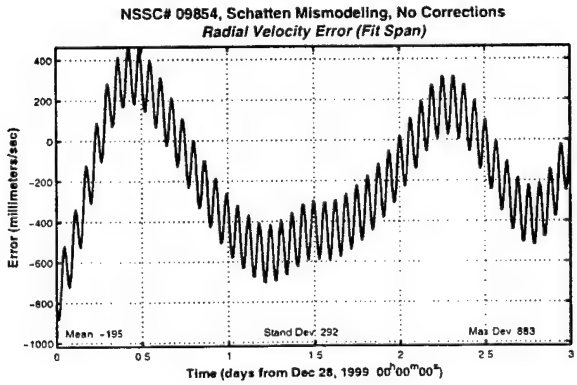












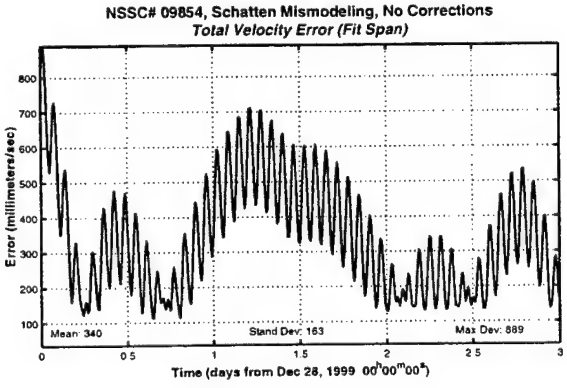


Figure A.13: NSSC# 09854 Predict Span Error, No Corrections

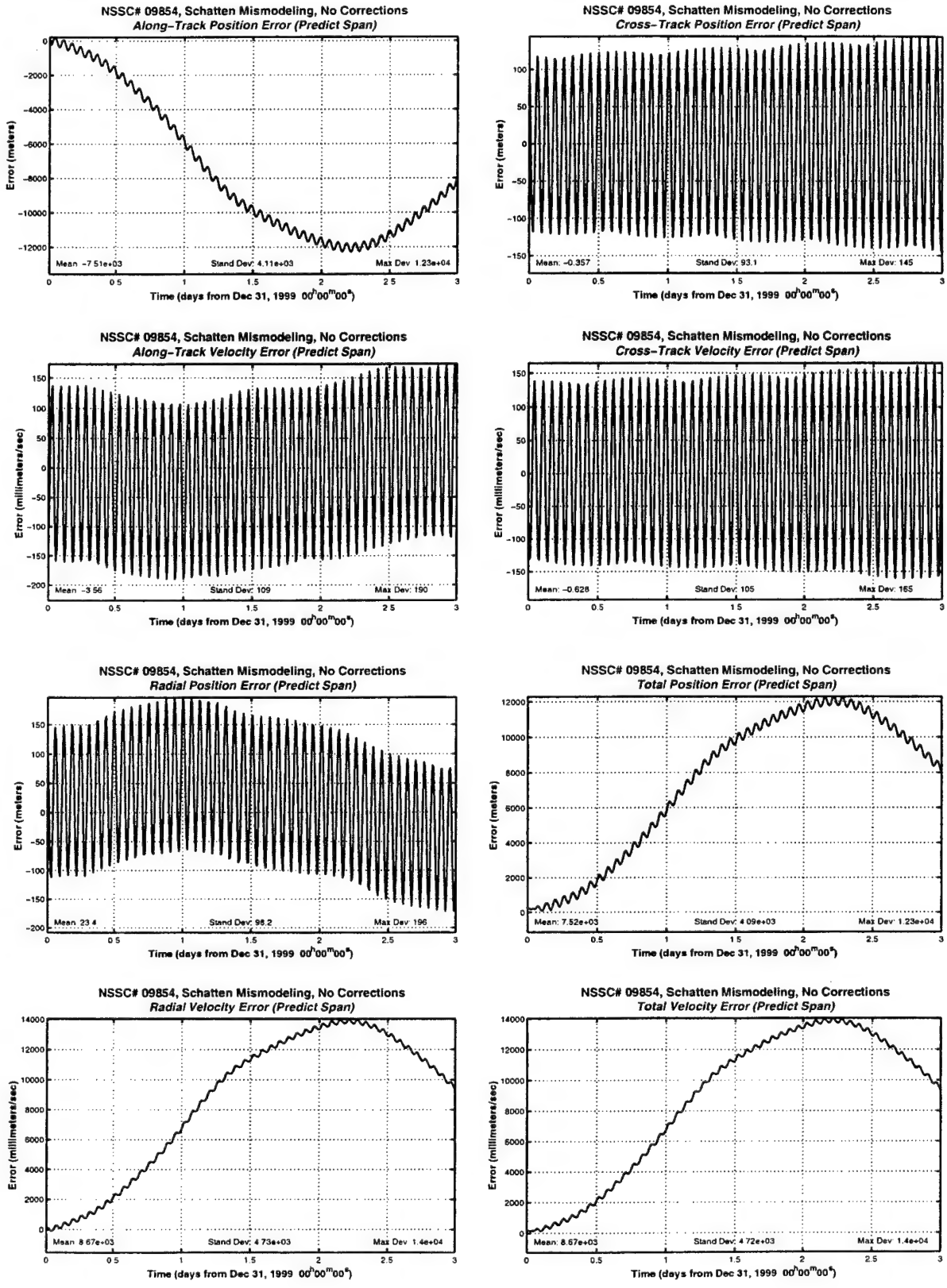


Figure A.14: NSSC# 09854 Fit Span Error, With Corrections

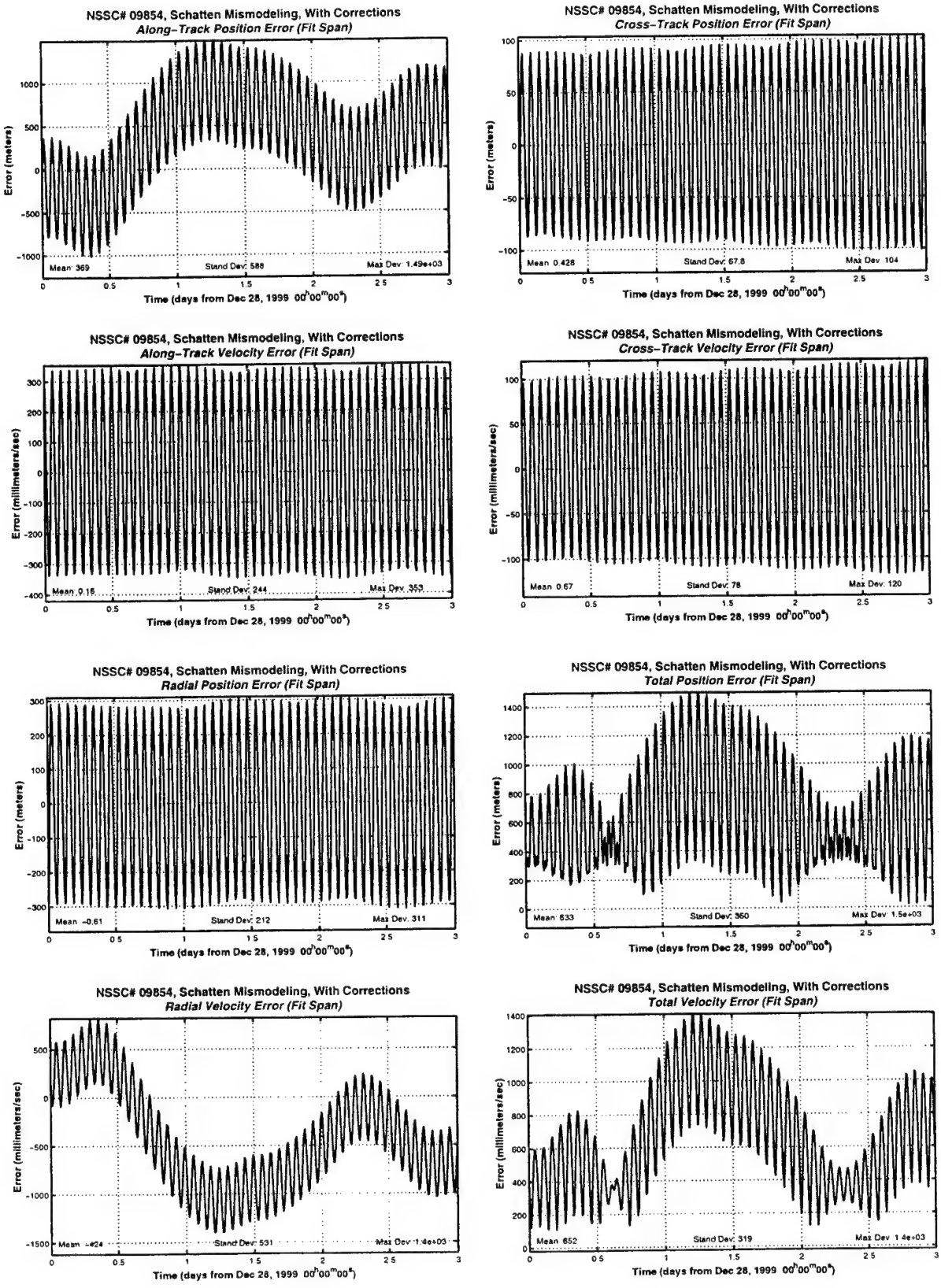


Figure A.15: NSSC# 09854 Predict Span Error, With Corrections

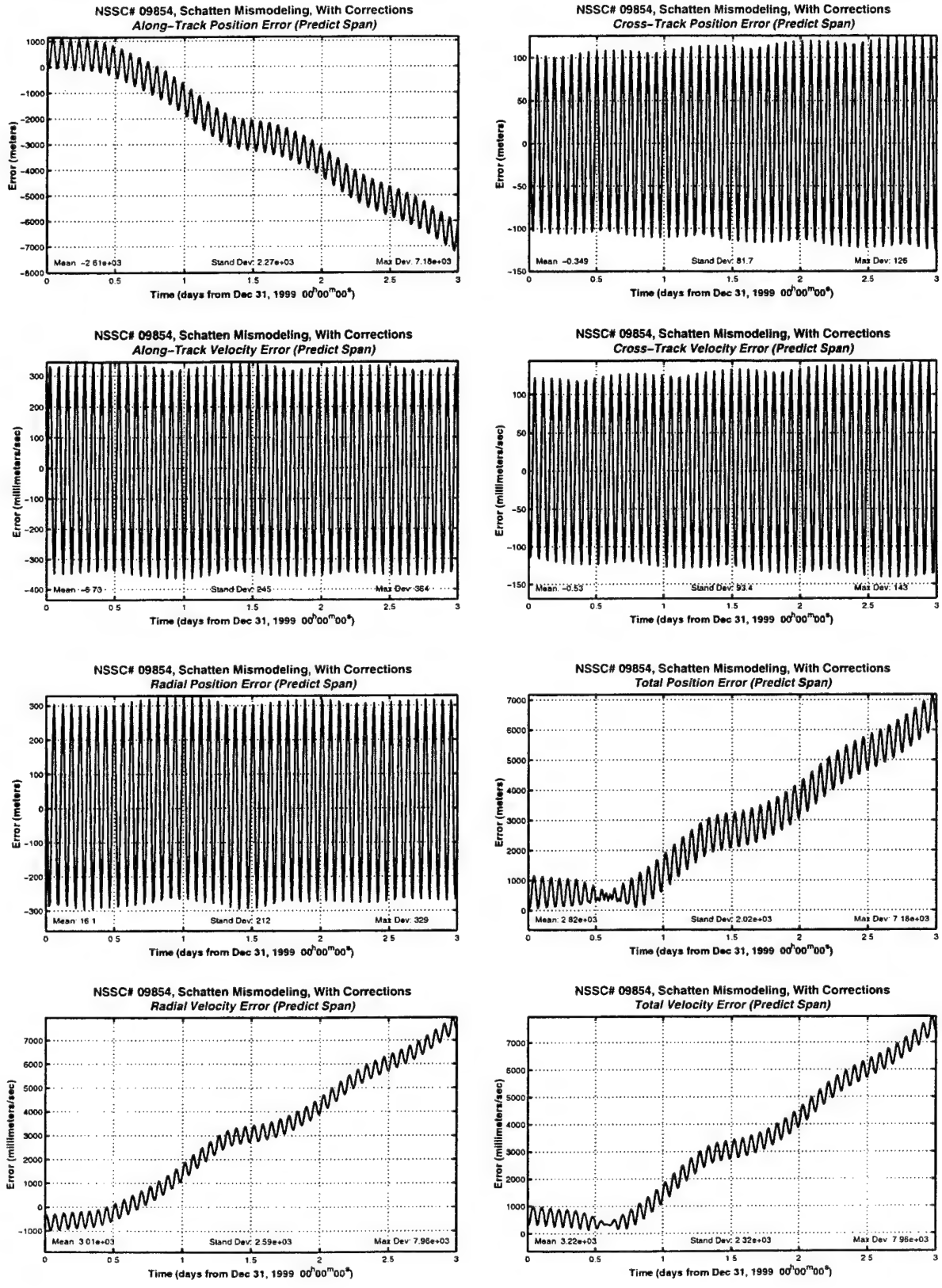


Figure A.16: NSSC# 17769 Fit and Predict Span Error, Without/With Corrections

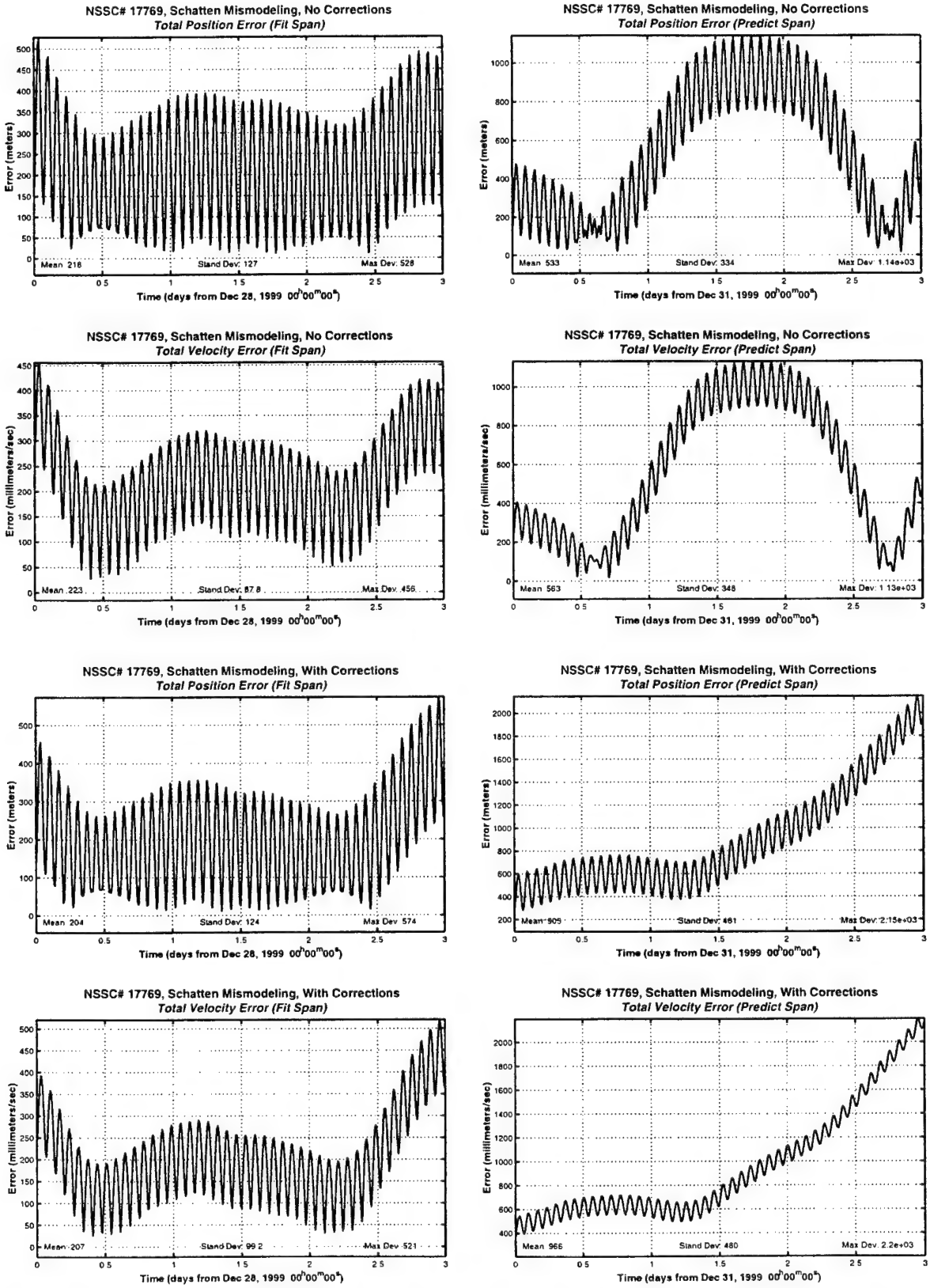
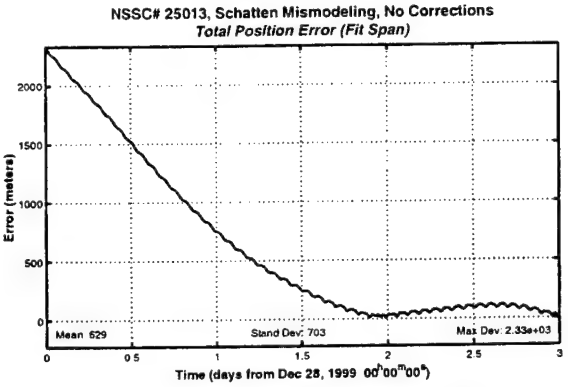
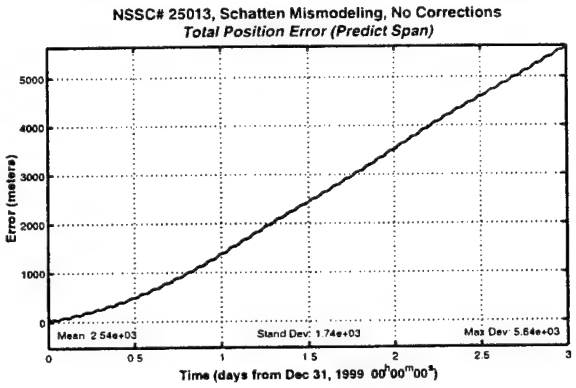
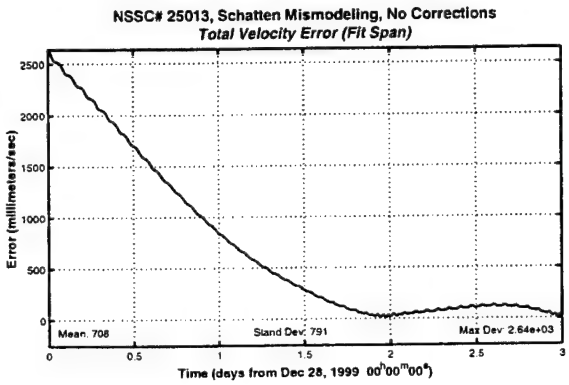
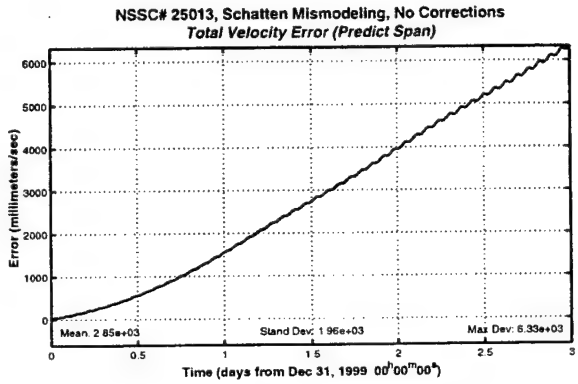


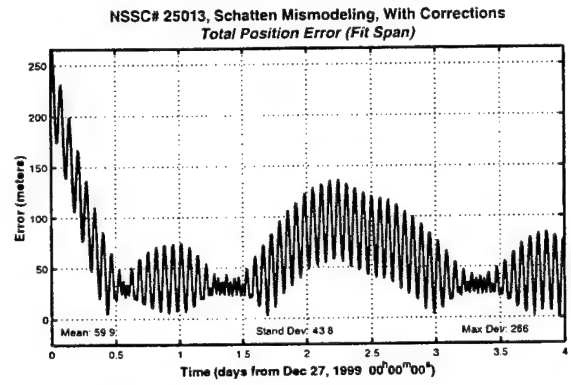
Figure A.17: NSSC# 25013 Fit and Predict Span Error, Without/With Corrections

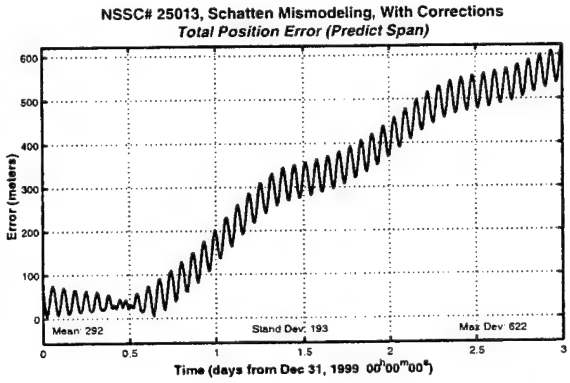


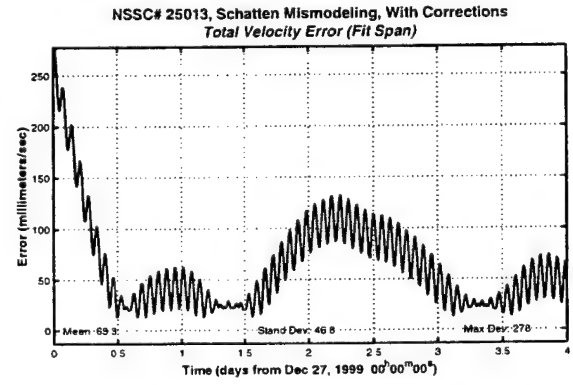












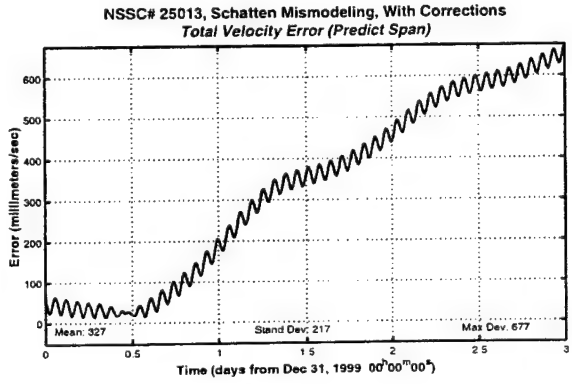
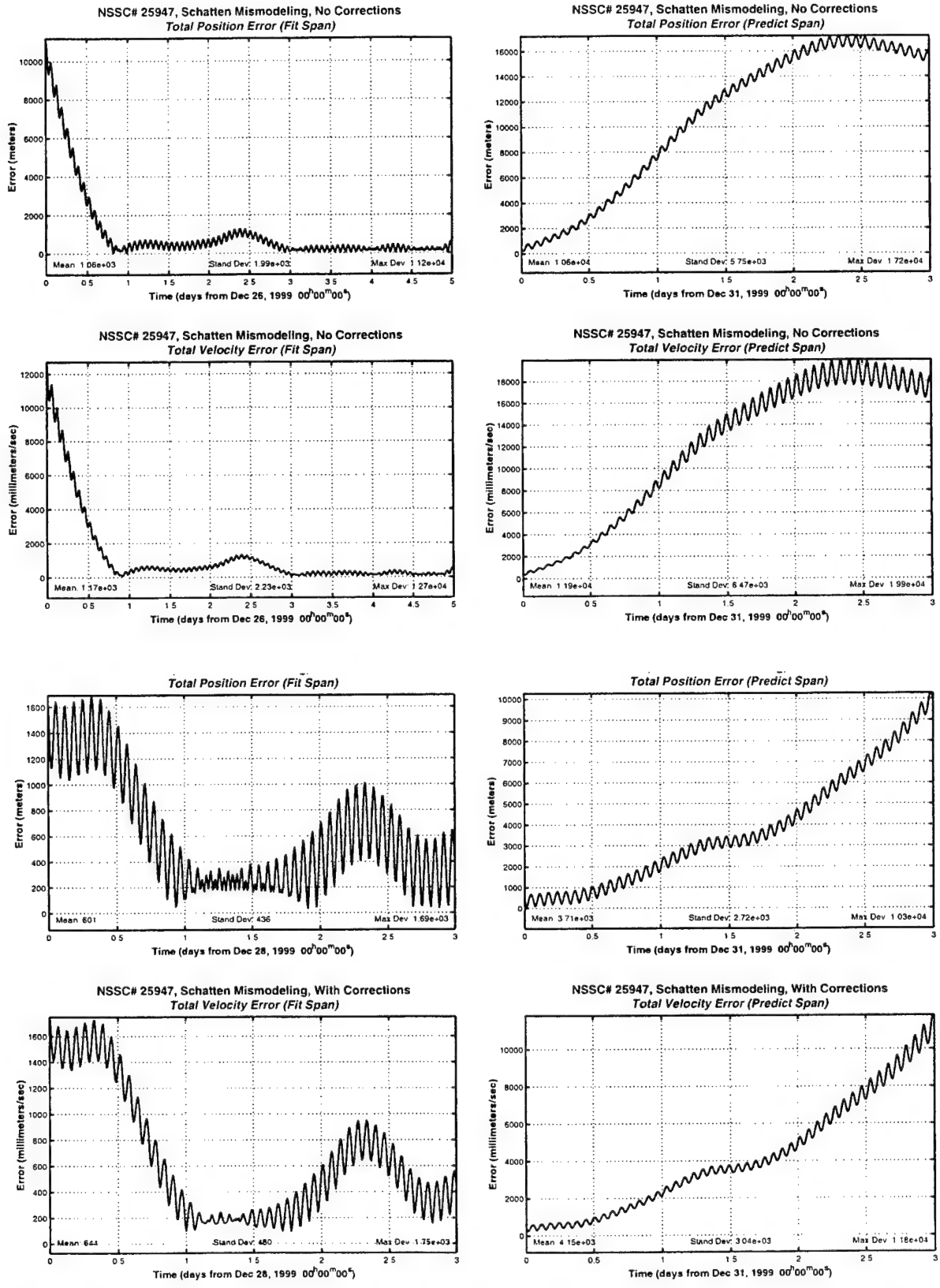


Figure A.18: NSSC# 25974 Fit and Predict Span Error, Without/With Corrections



[This Page Intentionally Left Blank]

Appendix B Use of UNIX-GTDS on DC1

This Appendix will serve as a reference for all considerations of running UNIX-GTDS on the DC1 SGI workstation. Sections include how to obtain and change source code with CVS; a description of the necessary data files and how to build them; instructions on compiling the code into an executable; and how to link to the appropriate data files and run a GTDS job. The final section is a description of known bugs or limited functionality to be addressed in the future.

B.1 Working with the Concurrent Versions System (CVS) 1.10.5

B.1.1 Introduction to Using the GTDS Libraries

Files incorporated into CVS are stored in "repositories", which contain read-only copies of the files in a format accessible by the CVS program. The files in the repositories are never changed directly, but must be checked in and out using CVS commands. The root of the repository can be defined in a UNIX environment variable called `CVSROOT`. This variable should be defined for GTDS users with the following line in the user's '`.cshrc`' file (usually located in the root or login directory):

```
setenv CVSROOT /usr/people2/realastro/cvsroot
```

All CVS repositories henceforth will be defined relative to this path. GTDS-related files are stored in the `cvsroot/gtds` repository, and future projects may be added in other repositories as desired.

Normally, if a user desires to modify code under CVS management, he or she must check out a working copy of the entire directory. However, this feature is rather inconvenient when dealing with the GTDS source code, since a complete working source directory contains more than 1200 files. Therefore, several new commands have been defined to allow individual checkouts of files from specific GTDS repositories.. These commands are summarized in Table B.1 below:

Table B.1: GTDS-Specific CVS Commands

Command	*Purpose*
`fetch filename`	Check out `filename` into current directory
`set_gs`	Set `gtds/source` as current repository Contains main GTDS source files
`set_gi`	Set `gtds/include` as current repository Contains all include files with '`.cmn`' extension
`set_gb`	Set `gtds/build_data` as current repository Contains source files, '`.com`' files, and text files needed to build GTDS binary files
`set_ge`	Set `gtds/exe` as current repository Contains files needed to compile and run GTDS
`set_gd`	Set `gtds/data` as current repository Contains GTDS binary files

The `fetch` command retrieves an individual file from the current repository into the working directory. The repository must first be specified with one of the '`set_gx`' commands. Five repositories have been defined under the central GTDS repository, and are described in Table B.1. above. To give an example, if a user wants to retrieve a copy of `aero.for` into the current directory:

```
dc1:1:~>set_gs
CVS Library = gtds/source
dc1:2:~>fetch aero.for
U ./aero.for
dc1:3:~>
```

If modifications are made to the code and the user wants to check the code back into the repository, he or she will type:

```
dc1:3:~>cvs commit -m "Included myheader.cmn" aero.for
Checking in aero.for;
/usr/people2/realastro/cvsroot/gtds/source/aero.for,v  <--  aero.for
new revision: 1.8; previous revision: 1.7
done
```

The files under CVS are each stored with a revision number, such as numbers 1.7 and 1.8 seen in the above example. It is possible to obtain or revert to old revisions of files; see Section B.1.2 for more details.

The primary archives are stored in various repositories under the CVSROOT directory, but another permanently checked-out copy of each repository is kept in a "library". Every time a change is committed to a file in the repository, the corresponding

file in the library is updated as well. The `gtds/source` repository has two associated libraries, `gtds/source` and `gtds/source_dbg`. This means that any change to a source file in the `gtds/source` repository is immediately reflected in *both* libraries, ensuring that the debug and non-debug libraries always have the same copies of source code. The libraries are used for compilation and execution, but again should not be accessed directly. The libraries are stored under the `/usr/people2/realastro/gtds` directory. The overall structure of the file storage is illustrated in Figure B.1 below:

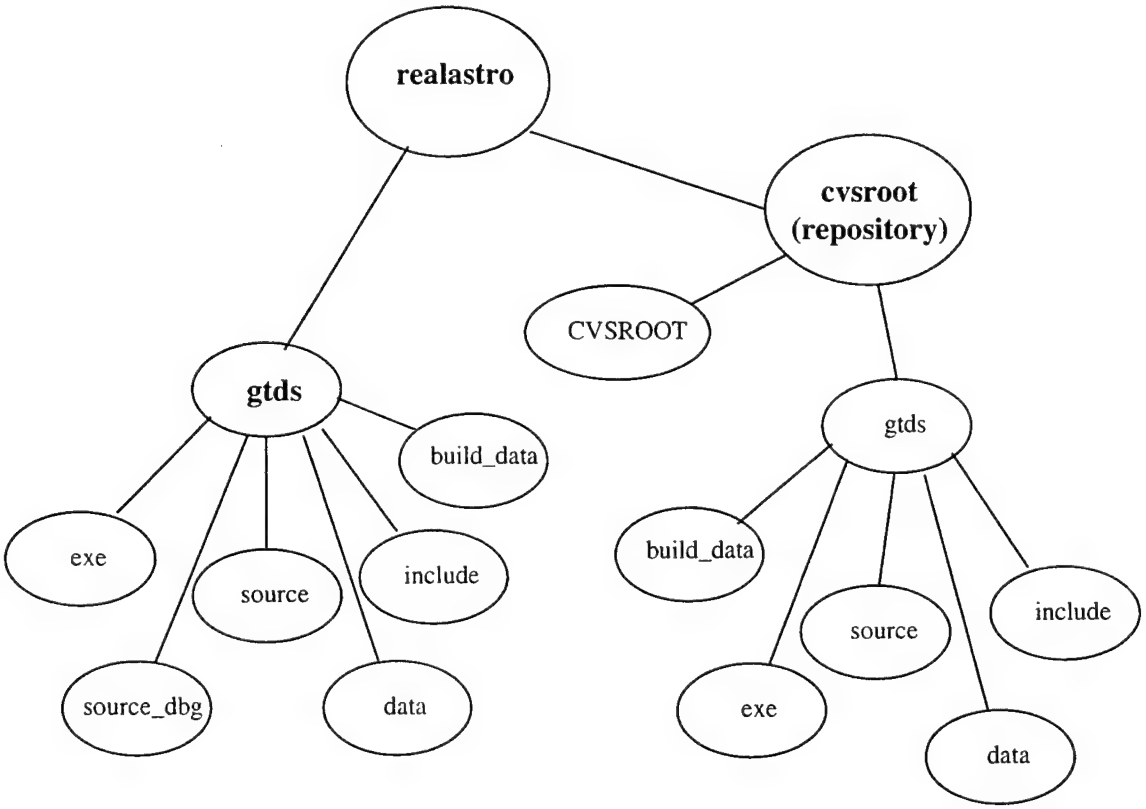


Figure B.1: The GTDS File Libraries and Repositories

All GTDS libraries on the left are paralleled by repositories on the right, with the exception of the `source_dbg` library, as this library contains the same source modules as the `source` library (but different object files). The only times a user should enter the library directories is when adding a new source file, or executing the routines for building

new data files. See Section B.1.3 for instructions on adding new source code, and Section B.2 on the creation of GTDS data files.

A final note is necessary about checking out source code from different repositories. CVS will not function properly in this situation unless all traces of the previous checkout have been removed from the working directory. CVS creates a "CVS" directory each time a file is checked out, which contains such information as where the repository is located and names of all checked-out files. After the file is checked in or deleted and the user wishes to fetch a file from a different repository, the CVS directory must be removed from the working directory or the new checkout will not be allowed.

B.1.1 Frequently-Used CVS Commands

Along with the commands listed in Table B.1, there are a number of commands that users will use frequently when examining or modifying GTDS source code. For further details on these commands and on all aspects of the CVS program, please refer to documentation by Cederqvist et. al. [54]. The commands are as follows:

Table B.2: Frequently Used CVS Commands

Command	*Purpose*
`cvs commit -m "comment" filename`	Commit *filename* to current GTDS repository with *comment*
`cvs add -m "comment" filename`	Adds *filename* to current GTDS repository with *comment* The *filename* to be added must be copied into the appropriate library
`cvs update -jnew_rev -jold_rev filename`	Reverses all changes between *new_rev* and *old_rev* in working copy of *filename*. Use `cvs commit` to officially check in *old_rev* in the repository.
`cvs history [options] filename`	Shows repository access history.

B.1.2 How to Add New Files to the GTDS Libraries

There are a number of steps that must be taken when incorporating new source code or files into the GTDS repositories. The easiest way to show these steps is through an example. Let us say that we are working on a new header file by the name of

myheader.cmn. We finish work on the file in the local directory and decide that we are ready to incorporate into GTDS. The first step is to set the appropriate repository:

```
dc1:4:~>set_gi
CVS Library = gtds/include
dc1:5:~>
```

Next, we must copy the file to be added into the library directory (not the repository!) and go to that location:

```
dc1:5:~>cp myheader.cmn /usr/people2/realastro/gtds/include
dc1:6:~>cd /usr/people2/realastro/gtds/include
dc1:7:~>
```

We are now ready to add the file:

```
dc1:7:~>cvs add -m "Initial importation" myheader.cmn
cvs1.10.5 add: scheduling file `myheader.cmn' for addition
cvs1.10.5 add: use 'cvs1.10.5 commit' to add this file permanently
dc1:8:~>cvs commit -m "Initial importation" myheader.cmn
RCS file: /usr/people2/realastro/cvsroot/gtds/include/myheader.cmn,v
done
Checking in myheader.cmn;
/usr/people2/realastro/cvsroot/gtds/include/myheader.cmn,v  <--
myheader.cmn
initial revision: 1.1
done

dc1:9:~>
```

The initial revision of myheader.cmn has been added to the gtds/include repository, and is automatically created in the appropriate library. **Note: New GTDS source code MUST be added from the gtds/source library.**

B.1.3 Setting Up the GTDS/CVS Environment

To use some of the commands described above and to tell other programs where to look for certain files, the following lines must be added to the user's '.cshrc' file:

```
# GTDS/CVS Environment Variables

setenv CVSROOT /usr/people2/realastro/cvsroot
setenv CVSEDITOR emacs
setenv GTDS_LIB /usr/people2/realastro/gtds/lib
setenv GTDS_DATA /usr/people2/realastro/gtds/data
setenv GTDS_EXE /usr/people2/realastro/gtds/exe
setenv STORAGE /pre3/granholm
setenv ATM_CAL $HOME/thesis/atm_cal
setenv ATM_EPHEM $STORAGE/ephem_runs
setenv ATM_DATASIM $STORAGE/datasim_runs
setenv ATM_DC $STORAGE/dc_runs

# GTDS/CVS Aliases

alias cvs           '/usr/local/bin/cvs1.10.5'
alias fetch         'cvs checkout -d . $CVSMODULE/\!^'
alias set_gs 'setenv CVSMODULE gtds/source;     echo "CVS Library = $CVSMODULE"';
alias set_gi 'setenv CVSMODULE gtds/include;    echo "CVS Library = $CVSMODULE"';
alias set_gb 'setenv CVSMODULE gtds/build_data; echo "CVS Library = $CVSMODULE"';
alias set_ge 'setenv CVSMODULE gtds/exe;        echo "CVS Library = $CVSMODULE"';
alias set_gd 'setenv CVSMODULE gtds/data;       echo "CVS Library = $CVSMODULE"';
alias make_gtds     'cd $GTDS_EXE; make; cd -'
alias make_gtds_dbg 'cd $GTDS_EXE; make -f "Makefile_dbg"; cd -'
```

Figure B.2: GTDS/CVS Modifications to .cshrc File

Note that a few of the environment variables relate specifically to atmospheric density correction (those variables beginning with "ATM") and can be left out if desired. A copy of a '`.cshrc`' file containing the above lines may be found on DC1 in `/usr/people/grg1787`.

B.2 The GTDS Data Files

The `gtds/data` library contains all of the necessary text and binary data files for execution of GTDS runs. Most of the files in this directory should not need to be recreated in the near future, but if such a situation does arise the necessary build files are located in the `gtds/build_data` library. All data files are kept under CVS management. The following files currently reside in `gtds/data`:

Table B.3: UNIX-GTDS Database Files

GTDS Logical Name	*File Name (*.dat)*	*Function*	*Effective Time Span*
GTDS$001	sfdir	stub associated with small files directory	
GTDS$002	atmosden	Harris-Priester atmosphere density tables	
GTDS$008	radarsat_earthfld	Earth Geopotential Field (21 x 21 models) updated for use in the Radarsat FD Program 1 = GEM T3 2 = GEM 10B 3 = WGS 84 (21 x 21 created from 12 x 12) 4 = JGM 2 5 = JGM 2 Clone 6 = WGS 72 (12 x 12) 7 = WGS 72 (12 x 12)	
GTDS$008	old_earthfld	The baseline 21 x 21 gravity models file 1 = SAO 1969 Standard Earth Model 2 = Earth Potential for Manned Flight Computations (EPMFC) 3 = GSFC Earth Model (GEM 1) 4 = GSFC Earth Model (GEM 7) 5 = GSFC Earth Model (GEM 9) 6 = GSFC Earth Model (GEM 10B truncated) 7 = WGS 72 (12 x 12) 8 = GSFC Earth Model (GEM L2 truncated) 9 = WGS 84 (12 x 12)	

GTDS$013	errormsg	GTDS Error Messages	
GTDS$014	gtds.de96.slp1950.bin.data	SLP Mean of 1950 file supplied by GSFC and used in Metzinger test cases	1 JAN 1974 to 17 JAN 1986
GTDS$014	june94.msgen.slp.mn1950	SLP Mean of 1950 file supplied by GSFC in June 94	24 DEC 1987 to 18 DEC 2007
GTDS$023	newcomb	Modified Newcomb Operator File. Designed for general use. Power of e (LTS) = 20, size of gravity field (NTS) = 21, power of e^2 (LHAN) = 10. If modified change size of COMMON block in nukes.cmn	
GTDS$038	gtds.de96.timecoef.bin	Timing Coefficient File supplied by GSFC and used for the Metzinger test cases	
GTDS$038	june94.msgen.slp.timcof	Timing Coefficient File supplied by GSFC in June 94	
GTDS$075	jacchia	Jacchia-Roberts Atmosphere Density Model used for the Metzinger test cases	2 MAR 1966 to 15 FEB 1986
GTDS$075	jrdat_nomn	Jacchia-Roberts Atmosphere Density Model data based on Ken Schatten's "nominal" solar activity and geomagnetic index predictions and "nominal" cycle timing; uses real data from 1980 – 1997	10 FEB 1980 to 30 SEP 2008
GTDS$075	jrdat_nomn_new	Jacchia-Roberts Atmosphere Density Model data based on Ken Schatten's "nominal" solar activity and geomagnetic index predictions and "nominal" cycle timing; uses real data from 1980 – 2000	10 FEB 1980 to 30 SEP 2008
GTDS$076	ms90_nomn	MSISE-90 Atmosphere Density Model data based on real NOAA data through February 1997, and on Ken Schatten's "nominal" solar activity and geomagnetic index predictions through September 2008. The file is for use with "nominal" cycle timing.	10 FEB 1980 to 30 SEP 2008

GTDS$078	gtds.de96.slptod.bin	SLP True of Date file obtained from the NASA GSFC and used in the Metzinger test cases	1 JAN 1974 to 17 JAN 1986
GTDS$078	june94.msgen.slp.tod 1950	SLP True of Date file supplied by GSFC in June 94	24 DEC 1987 to 18 DEC 2007
GTDS$106	jac_densvars_sn_all_ fc	Jacchia-Roberts Correction File used to correct the Schatten model with noise	17 DEC 1999 to 9 FEB 2000

Each of these files is built using a driver program which links the input and output files to the appropriate file names, compiles and runs the build code, and then cleans up and exits. An example of one such driver program is given in Figure B.4 below:

```
#-----------------------------------------------------------------------
#  WRITATM.COM   This .COM file builds the GTDS
#                Harris-Priester atmospheric density binary file
#               (linked to GTDS$002) from the
#               corresponding text file.
#-----------------------------------------------------------------------
#
    make writatm
#
#  Remove any existing links for files 'input' and 'output'
#
    rm input  >& /dev/null
    rm output >& /dev/null
#
#   Make links for our input and output files
#
   ln -s /usr/people2/realastro/gtds/build_data/atmosden.txt        input
   ln -s /usr/people2/realastro/gtds/data/atmosden.dat              output
#
#
    writatm.exe
#
    rm input  >& /dev/null
    rm output >& /dev/null
#
#-----------------------------------------------------------------------
```

Figure B.4: Example of Data File Driver Program

If a user desires to build a new data file, it is not necessary to change any of the source code unless functional modifications are desired. The user must simply set the

input and output links in the appropriate `.com` file and execute. Table B.3 outlines the various driver programs and the corresponding GTDS data files.

Table B.3: GTDS Data File Driver Programs

Filename	*GTDS Data File*
`jrmsis2.com`	Jacchia-Roberts '71 Atmospheric Density File (GTDS$075)
`jrmsisin.com`	MSISE-90 Atmospheric Density File (GTDS$076)
`jrmsisrl.com`	Intermediate text files from real NOAA data to be used by `jrmsisin.com` or `jrmsis2.com`
`jrmsissp.com`	Intermediate text files from predicted Schatten data to be used by `jrmsisin.com` or `jrmsis2.com`
`writatm.com`	Harris-Priester Atmospheric Density File (GTDS$002)
`writerr.com`	GTDS Error Messages file (GTDS$013)
`writhrm.com`	Gravitational Potential file (GTDS$008)
`writion.com`	Ionospheric Refraction file (GTDS$039) – not currently used
`writnuk.com`	Newcomb Coefficients file (GTDS$023). Input options may be set in `newcomb.txt`
`writsfd.com`	Small Files Directory file (GTDS$001)
`writslp.com`	Solar/Lunar/Planetary Ephemeris file, either mean of date (GTDS$014) or true of date (GTDS$078)
`writslr.com`	Solar Flux file (GTDS$059) – not currently used
`writtim.com`	Timing Coefficients file (GTDS$038)

If a user want to add a new data file into the CVS repository, the same procedures outlined in Section B.1.3 can be followed with one exception: the "-kb" option must be added to the `cvs add` command as follows:

```
dc1:7:~>cvs add -kb -m "Initial binary importation" newbinary.dat
```

This option tells the CVS program that the named file is in binary format, and will suppress line ending conversions and CVS keyword expansion.

B.3 Compiling, Linking, and Executing GTDS

The compilation of GTDS code has been simplified as much as possible using `Makefiles` and newly-defined commands such as `make_gtds`. Again, the best way to explain is through an example. Let us say that we have modified two files in the GTDS database: `aero.for` and `myheader.cmn`. We successfully added and checked

in the files, and are now ready to compile a new version of GTDS. We have not added any new source code (new header files do not require any modifications to the compilation scheme), so we may simply type:

```
dc1:9:~>make_gtds
```

The `make_gtds` command executes the `make` command, which in turn looks at the various source files specified in the `Makefile` to see if they need to be recompiled. In this case, `make` will see that `aero.for` is out-of-date and will recompile the source code into a new object file. Next, all object files will be archived into a library residing in the `gtds/lib` directory, and the object files are linked into a new executable in the `gtds/exe` directory. Part of the GTDS `Makefile` is shown in the figure below:

```
        # UNIX-GTDS PR5 Makefile
        # Vers. 2  Apr 29, 2000
        # Author: G. Granholm

        INC_FILE = /usr/people2/realastro/gtds/exe/head.mk
        include $(INC_FILE)

        # Block Datas

        GTDS_OBJS1 = \
        $(GTDS_SRC)/adconsbd.o \
        $(GTDS_SRC)/anavdpbd.o \
        $(GTDS_SRC)/anavinbd.o \
        $(GTDS_SRC)/anlfilbd.o \
        $(GTDS_SRC)/aprbd.o    \
        $(GTDS_SRC)/ascbd.o    \
        $(GTDS_SRC)/atmanibd.o \
        $(GTDS_SRC)/attobcbd.o \
        .
        .
        .
        $(GTDS_SRC)/errfunct.o   \
        $(GTDS_SRC)/gnef4.o      \
        $(GTDS_SRC)/seteskf.o    \
        $(GTDS_SRC)/ascii_orb1_data.o \
        $(GTDS_SRC)/initcaljac.o \
        $(GTDS_SRC)/calccaljac.o

        default: all

        all: archgtds linkgtds

        archgtds: $(GTDS_OBJS1) $(GTDS_OBJS2) $(GTDS_OBJS3) $(GTDS_OBJS4) $(GTDS_OBJS5) \
                  $(GTDS_OBJS6) $(GTDS_SRC)/odsexec.o
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS1)
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS2)
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS3)
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS4)
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS5)
                $(AR) -v $(GTDS_LIB)/libgtds.a $(GTDS_OBJS6)

        linkgtds: $(GTDS_OBJS)
                $(F77) $(FFLAGS) -o $(GTDS_EXE)/gtds.exe $(GTDS_OBJS1)
$(GTDS_SRC)/odsexec.o \
                 $(GTDS_LIB)/libgtds.a

        clean:
                - rm core *.o *.*L
```

Figure B.5: The UNIX-GTDS Makefile

If a new source file is to be added to GTDS, it is simply appended on the end of the list of object files in the `Makefile`. If the source file is a BLOCK DATA file, it must be added in the first group of objects denoted by `$GTDS_OBJS1`. If the new file is a header file, the `Makefile` does not have to be modified.

A separate compilation command (`make_gtds_dbg`) and Makefile (`Makefile_dbg`) are used for creating a debugging version of GTDS. The only real

difference is in the compilation options defined in the `make include` file, which is named `head.mk` or `head_dbg.mk` for the non-debugging or debugging versions of the code, respectively. Make sure to re-make both debugged and non-debugged versions of GTDS if the source code has been changed.

After the code has been compiled, we are ready to execute. Another driver program by the name of `run_gtds.com` has been created to standardize and simplify the way GTDS is executed. A sample `run_gtds.com` file is given in Figure B.6 below:

```
#-----------------------------------------------------------------------
#  RUN_GTDS.COM  This .COM file makes all necessary links
#                to data files, obs cards, and input and output
#               files, runs GTDS, removes links, and exits.
#
#-----------------------------------------------------------------------
#
#  Remove any existing links to GTDS$ files
#
   rm GTDS\$*  >& /dev/null
#
#  Make new links for binary files
#
   ln -s /usr/people2/realastro/gtds/data/sfdir.dat                GTDS\$001
   ln -s /usr/people2/realastro/gtds/data/atmosden.dat             GTDS\$002
   ln -s /usr/people2/realastro/gtds/data/radarsat_earthfld.dat    GTDS\$008
#   ln -s /usr/people2/realastro/gtds/data/old_earthfld.dat        GTDS\$008
   ln -s /usr/people2/realastro/gtds/data/errormsg.dat             GTDS\$013
   ln -s /usr/people2/realastro/gtds/data/gtds.de96.slp1950.bin.data GTDS\$014
   ln -s /usr/people2/realastro/gtds/data/newcomb.dat              GTDS\$023
   ln -s /usr/people2/realastro/gtds/data/gtds.de96.timecoef.bin.data GTDS\$038
#   ln -s /usr/people2/realastro/gtds/data/jacchia.data            GTDS\$075
   ln -s /usr/people2/realastro/gtds/data/jrdat_nomn.dat           GTDS\$075
#   ln -s /usr/people2/realastro/gtds/data/ms90_nomn.dat           GTDS\$076
   ln -s /usr/people2/realastro/gtds/data/gtds.de96.slptod.bin.data GTDS\$078
#
#  Call local .COM file to make input/output links
#
   source /usr/people/grg1787/thesis/gtds_tests/pr5/test5/test5.com
#
#  Run the executable
#
   echo ""
   echo ""
   echo  UNIX-GTDS
   echo  Charles Stark Draper Laboratory
   echo ""
   echo  Run started at:
   date
   /usr/people2/realastro/gtds/exe/gtds.exe
   echo  Run ended at:
   date
#
#  Remove links
#
    rm GTDS\$*  >& /dev/null
#
#-------------- ---------------------------------------------------------
```

Figure B.6: The `run_gtds.com` File

This driver program makes all the links to the necessary data files, calls a local driver program to make input/output links, and executes GTDS. If the code is to be executed with a debugger, `run_gtds_dbg.com` may be used instead. This driver file will run the code under the DBX Fortran debugger as a default.

All of the files discussed in Section B.3, including `Makefiles`, `include` files, and driver programs, are configuration managed in the `gtds/exe` repository.

B.4 List of Known GTDS Bugs and Functional Limitations

The following is a list of known bugs or functional limitations of the GTDS code as currently implemented on the UNIX system.

1) **Hang-up Error**: this error sometimes occurs when low-altitude objects are calculated to impact the Earth. GTDS appears to hang up and must be manually interrupted. A possible culprit is the SECHEK.FOR routine. This error may be reducing the number of runs that converge in the density correction process.
2) **DC Epoch Limitation**: When using an input .OBS file (GTDS$029) for a DC run, GTDS halts execution unless the start of the OBSINPUT card matches the solve-for EPOCH.
3) **Random Number Generation Bug**: There appears to be a bias in random noise added to observations using DATASIM only when the optimized compilation of GTDS is executed. If the non-optimized (debug) version of the code is used, the bias disappears. The source of the error appears to be RANDU.FOR.
4) **Y2K Bug in Station Pass Report**: The full date field does not appear for dates after Jan 1, 2000 in the DATASIM Station Pass Report.
5) **Residual Plot Error**: DC Residual plots are not functional.

Appendix C Density Correction Software

This Appendix presents the four Perl scripts and the Matlab file used for density corrections.

C.1 The TLE2osc.pl Program

```
#!/usr/bin/perl -I/usr/people/grg1787/thesis/atm_cal/include -w
#
# TLE2osc.pl - TLE Conversion Program
#
# Author:
#
# George R. Granholm
# 22 Mar 00
#
#

use Dates;       # Necessary to use cal2jul, jul2cal, & get_time subroutines
use FileHandle; # For autoflush

# Set options and variables

$start_epoch = "991215 000000.0";  # Must be at least 1 minute after last TLE epoch
$end_epoch   = "1000211 000000.0";
$model_opt   = "lowgrav";
$tle_file    = "$ENV{ATM_CAL}/200_600_tles.txt";
$logfile     = "$ENV{ATM_CAL}/${model_opt}/TLE2osc.log";
$initfile    = "$ENV{ATM_CAL}/${model_opt}/initinfo.txt";
$rcsfile     = "$ENV{ATM_CAL}/rcs.txt";
$time_limit  = 180;

($start_ymd, $start_hms) = split(" ", $start_epoch);
($end_ymd, $end_hms) = split(" ", $end_epoch);

# Open necessary files

open LOGINFO, ">>$logfile";
open STDERR, ">>&LOGINFO";
open TLES, $tle_file or die "Invalid TLE filename: $!\n";
open INITINFO, ">>$initfile";

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "INITINFO") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tTLE2osc.pl: Processing $tle_file\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Read in RCS into $rcs{$catnum} hash

open RCSFILE, "<$rcsfile";
while (defined($rcsline = <RCSFILE>)) {
```

```
    chop $rcsline;
    ($catnum, $area) = split(" ", $rcsline);
    $catnum = sprintf "%5.5d", $catnum;
    if ($area) { $rcs{$catnum} = $area;}
    else { $rcs{$catnum} = 2.2;}   # Set default to 2.2 m^2 if no data
}
close RCSFILE;

# Read from TLE file

 LINE: while (defined($line = <TLES>)) {   # Main loop through TLE file

     next LINE if ($line =~ /^[^12][^\s][^\d]/);
     if ($line =~ s/^1\s(\d{5})\w\s(\d{5})\s*(\w{1,3})//) {   #  Match first TLE line
        $catnum = $1;
        $intl_des = $2 . $3;
        chop $line;
        @line = split(" ",$line);
        $norad_date = $line[0];

        # Convert NORAD epoch to calender date

        ($yr, $day) = ($norad_date =~ /(^\d{2})(\d{3}\.\d{8})/);
        $yr_days = cal2jul($yr,1,1,0,0,0);          # First convert year to Julian date
        $nor_juldat = ($yr_days + $day - 1);        # Add day number to Julian date
        @nor_caldat = jul2cal($nor_juldat);         # Convert back to calender date
        ($nor_caldat[0]) = ($nor_caldat[0] =~ /\d{2}(\d{2})/);    # Two-digit year
        if ($nor_caldat[0] == 0) {$nor_caldat[0] = "100";}        # GTDS Y2K fix
        $ymd = join("",@nor_caldat[0 .. 2]);
        $hms = join("",@nor_caldat[3 .. 5]);

        # Calculate end time of GP4 propagation (one minute after NORAD epoch)

        $gp4end_jul = $nor_juldat + 1/1440;     # Next minute after NORAD epoch
        @gp4end_cal = jul2cal($gp4end_jul);
        ($gp4end_cal[0]) = ($gp4end_cal[0] =~ /\d{2}(\d{2})/);    # Two-digit year
        if ($gp4end_cal[0] == 0) {$gp4end_cal[0] = "100";}        # GTDS Y2K fix
        $gp4end_ymd = join("",@gp4end_cal[0 .. 2]);
        $gp4end_hms = join("",@gp4end_cal[3 .. 5]);

        # Read remaining elements

        $dndt = $line[1];
        $d2ndt2 = $line[2];
        $bstar = $line[3];

        # Convert d2n/dt2 and B* to standard numerical formats

        if ($d2ndt2 =~ /(-*)(\d{5})([+-]\d)/) {
            $d2ndt2 = $1 . "0." . $2 . "E" . $3;
        }

        if ($bstar =~ /(-*)(\d{5})([+-]\d)/) {
            $bstar = $1 . "0." . $2 . "E" . $3;
        }

        # Apply Dave Vallado's multiplier to obtain B from B*, and
        # compute drag coefficient using RCS area and default C_d

        $ball_fact = 6.3708105*$bstar;       # where B = 1/2 (Ax/m) C_d
        $Ax = $rcs{$catnum};                 # in m^2
        $C_d = 2.2;                          # Default LEO C_d
        $mass = ($Ax*$C_d)/(2*$ball_fact);  # in kg
         $Ax_km = sprintf("%7.10E",($Ax/1000000));          # Convert to km^2

     }

     elsif ($line =~ /^2\s(\d{5})/) {   #  Match second TLE line
        if ($bstar == 0 ) {next LINE};
        chop $line;
        @line = split(" ",$line);
```

```
$incl = $line[2];
$raan = $line[3];
$ecc = $line[4];
$aop = $line[5];
$ma = $line[6];
$mm = $line[7];

# Convert eccentricity to standard numerical format

if ($ecc =~ /(\d{7})/) {
    $ecc = "0." . $1;
}

# Separate mean motion from rev number if necessary

if ((length($mm) > 11) && ($mm =~ /(\d{1,2}\.\d{8})/)) {
    $mm = $1;
}

# goto WRITEINFO;   # If you only want to generate initinfo.txt

# Write GTDS card file

$ephem_card  = "${catnum}_ephem.gtds";
$output_file = "${catnum}_ephem.output";
$orbit_file  = "${catnum}_ephem.orbit";
$orb1_file   = "${catnum}_ephem.orb1";
$ascii_file  = "${catnum}_ephem.ascii";

open(EPHEM_CARD, ">$ENV{ATM_EPHEM}/${model_opt}/$ephem_card");
write EPHEM_CARD;
close EPHEM_CARD;

# Make standard data file links

system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$* fls
symlink("$ENV{GTDS_DATA}/sfdir.dat",                      "GTDS\$001");
symlink("$ENV{GTDS_DATA}/atmosden.dat",                   "GTDS\$002");
symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",          "GTDS\$008");
symlink("$ENV{GTDS_DATA}/errormsg.dat",                      "GTDS\$013");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat",    "GTDS\$014");
symlink("$ENV{GTDS_DATA}/newcomb.dat",                    "GTDS\$023");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat",    "GTDS\$038");
symlink("$ENV{GTDS_DATA}/jrdat_nomn_new.dat",             "GTDS\$075");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat",   "GTDS\$078");

# Make job-specific data links

symlink("$ENV{ATM_EPHEM}/${model_opt}/$ephem_card",   "GTDS\$005");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$output_file", "GTDS\$006");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$orbit_file",   "GTDS\$020"); # GTDS storage
symlink("$ENV{ATM_EPHEM}/${model_opt}/$orb1_file",    "GTDS\$024");
symlink("$ENV{ATM_EPHEM}/${model_opt}/$ascii_file",   "GTDS\$101");

# Run GTDS!

foreach $fh ("STDOUT","LOGINFO") {
    print $fh "-" x 40,"\n";
    print $fh "  Processing NORAD Catalog \#$catnum\n";
    print $fh "-" x 40,"\n";
    print $fh "UNIX-GTDS\n";
    print $fh "Charles Stark Draper Laboratory\n\n";
    print $fh ("Run started at: ", get_time(), "\n");
}

undef $child_id;

if ($child_id = fork) {          # Parent process here

    local $SIG{USR1} = sub {    # Define anonymous sub to kill GTDS
```

```
            (my $gtds_id) = split (" ", `ps | grep gtds`);

            foreach $fh ("STDOUT","LOGINFO") {
                print $fh "GTDS run has exceeded time limit;\n";
                print $fh "Killing process $gtds_id\n";
            }

            kill 'QUIT', $gtds_id;
        };
        waitpid $child_id, 0;       # Wait for child process to finish
    }

    elsif (defined $child_id) {     # Child process here

        $par_id = getppid;

        local $SIG{ALRM} = sub {    # Define local ALRM signal handler
            kill 'USR1', $par_id;  # Send USR1 signal to parent if lcl alrm goes off
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh "Sending USR1 to $par_id..\n";
            }
        };

        alarm $time_limit;          # Initialize alarm to go off in $time_limit sec
        system("$ENV{GTDS_EXE}/gtds.exe");
        alarm 0;                    # Turn off alarm if finish before $time_limit
        die "Exiting child process..."
    }

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Run ended at: ", get_time(), "\n");
    }

    # Compress output files using gzip

    foreach $fh ("STDOUT","LOGINFO") {
        print $fh ("Compressing .orbit file...\n");
    }

    system "gzip -v -f $ENV{GTDS_STOR}/${model_opt}/$orbit_file";
    system "gzip -v -f $ENV{ATM_EPHEM}/${model_opt}/$output_file";

    system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$* fls
    system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

    # Write line in INITINFO array

  WRITEINFO: {

      $stan_flag = 'S';    # All satellites are standard
      $var = 1E-6;         # Default variance for standard satellites
      $obs_type = 29;      # Obs type for simulated observations
      printf INITINFO "%5s %8s %7.10E %7.10E %7.10E %1s %2d\n", $catnum, $intl_des
          , $ball_fact, $Ax, $var, $stan_flag, $obs_type;
  }

 }

}

close TLES;
close INITINFO;
close LOGINFO;

#==============   EPHEM card deck formatting ====================================

format EPHEM_CARD =
CONTROL   EPHEM                                              @<<<<<<<  @>>>>>>>
                                                             $intl_des, $catnum

EPOCH             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                  $ymd,                $hms
```

```
ELEMENT1   8 18   1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $mm,                  $ecc,                 $incl
ELEMENT2            @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $raan,                $aop,                 $ma
ELEMENT3            @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $dndt,                $d2ndt2,              $bstar
OUTPUT     1   2   1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 60.0
                    $gp4end_ymd,          $gp4end_hms
ORBTYPE   14   1   8  1
OGOPT
POTFIELD   1   7
END
FIN
CONTROL    EPHEM                           OUTPUT                @<<<<<<<  @>>>>>>>
                                                                 $intl_des, $catnum
OUTPUT     1   2   1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 86400.0
                    $start_ymd,           $start_hms
ORBTYPE    2   1   1  60.0
OGOPT
ATMOSDEN           1
DRAG       1        1
DRAGPAR    3   0    @<<<<<<<<<<<<<<<<<<<
                    $C_d
SCPARAM             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $Ax_km,               $mass
POTFIELD   1   4
MAXDEGEQ   1         4.0
MAXORDEQ   1         4.0
SOLRAD     1         1.0
END
FIN
CONTROL    EPHEM                           OUTPUT                @<<<<<<<  @>>>>>>>
                                                                 $intl_des, $catnum
OUTPUT     1   2   1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 86400.0
                    $end_ymd,             $end_hms
ORBTYPE    2   1   1  60.0
OGOPT
ATMOSDEN           1
DRAG       1        1
DRAGPAR    3   0    @<<<<<<<<<<<<<<<<<<<
                    $C_d
SCPARAM             @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $Ax_km,               $mass
POTFIELD   1   4
MAXDEGEQ   1         4.0
MAXORDEQ   1         4.0
SOLRAD     1         1.0
OUTOPT     2   2   1 @>>>>>>@<<<<<<<<<<<< @>>>>>>@<<<<<<<<<<<< 600
                    $start_ymd, $start_hms, $end_ymd, $end_hms
END
FIN
.
```

C.2 The genobs.pl Program

```
#!/usr/bin/perl -I/usr/people/grg1787/thesis/atm_cal/include -w
#
# genobs.pl - Observation Generator Program
#
# Author:
#
# George R. Granholm
# 06 Apr 00
#
#

use Dates;       # For cal2jul, jul2cal, & get_time subroutines
```

```
use Localmath;   # For round subroutine
use FileHandle;  # For buffer autoflush

# Set variables and options

$start_epoch = "991215 000000.0";
$end_epoch   = "1000211 000000.0";
$ephem_opt   = "lowgrav";
$datasim_opt = "lowgrav_noise";
$logfile     = "$ENV{ATM_CAL}/${datasim_opt}/genobs.log";
$initfile    = "$ENV{ATM_CAL}/${datasim_opt}/initinfo.txt";
$time_limit  = 180;
*PI          = \3.14159265358979;

# Define hash which contains obs types

%obstype = (
           RANG => 1,
           AZ   => 4,
           EL   => 5,
           );

# Format start epoch

($start_ymd, $start_hms) = split(" ", $start_epoch);
$start_ymd2 = $start_ymd;
if (length($start_ymd2) == 7) {
    ($start_ymd2) = ($start_ymd2 =~ /(\d{6})$/);          # Take off GTDS Y2K fix
}                                                         # for Julian date conversion
($y,$m,$d) = ($start_ymd2 =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($start_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$start_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Calculate interval times for tracking schedule

$end_interval1 = $start_jul + 1/4;    # Six hours after start
$end_interval2 = $start_jul + 2/4;    # Twelve hours after start
$end_interval3 = $start_jul + 3/4;    # Eighteen hours after start
$end_interval4 = $start_jul + 1;    # Twenty-four hours after start
@interval1 = jul2cal($end_interval1);
@interval2 = jul2cal($end_interval2);
@interval3 = jul2cal($end_interval3);
@interval4 = jul2cal($end_interval4);
($interval1[0]) = ($interval1[0] =~ /\d{2}(\d{2})/);     # Two-digit year
if ($interval1[0] == 0) {$interval1[0] = "100";}         # GTDS Y2K fix
($interval2[0]) = ($interval2[0] =~ /\d{2}(\d{2})/);     # Two-digit year
if ($interval2[0] == 0) {$interval2[0] = "100";}         # GTDS Y2K fix
($interval3[0]) = ($interval3[0] =~ /\d{2}(\d{2})/);     # Two-digit year
if ($interval3[0] == 0) {$interval3[0] = "100";}         # GTDS Y2K fix
($interval4[0]) = ($interval4[0] =~ /\d{2}(\d{2})/);     # Two-digit year
if ($interval4[0] == 0) {$interval4[0] = "100";}         # GTDS Y2K fix

$interval1_ymdhms = join("",@interval1);
$interval2_ymdhms = join("",@interval2);
$interval3_ymdhms = join("",@interval3);
$interval4_ymdhms = join("",@interval4);

# Format end epoch

($end_ymd, $end_hms) = split(" ", $end_epoch);
$end_ymd2 = $end_ymd;
if (length($end_ymd2) == 7) {
    ($end_ymd2) = ($end_ymd2 =~ /(\d{6})$/);  # Take off GTDS Y2K fix
}
($y,$m,$d) = ($end_ymd2 =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($end_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$end_jul = cal2jul($y,$m,$d,$h,$mn,$s);

$span_len = round($end_jul - $start_jul);

# Open log file
```

```
open LOGINFO, ">>$logfile";
open STDERR, ">>&LOGINFO";

foreach $fh ("STDOUT", "LOGINFO", "STDERR") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tgenobs.pl: Processing $initfile\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Open and read $initfile

open INITINFO, "<$initfile" or die "Can't find $initfile";

 INITLINE: while (defined($line = <INITINFO>)) {

     $line =~ s/^(\d{5})\s//;
     $initinfo{$1} = [ split(" ",$line) ];
 }

close INITINFO;

# Begin main loop by $catnum

foreach $catnum (sort keys %initinfo) {

    foreach $fh ("STDOUT","LOGINFO") {
            print $fh "-" x 40,"\n";
            print $fh "  Processing NORAD Catalog \#$catnum\n";
            print $fh "-" x 40,"\n";
        }

    $intl_des = $initinfo{$catnum}[0];

    # Write GTDS card file

    $datasim_card = "${catnum}_datasim.gtds";
    $output_file  = "${catnum}_datasim.output";
    $orbit_file   = "${catnum}_ephem.orbit";
    $obs_file     = "${catnum}_datasim.obscard";

    open(DATASIM_CARD, ">$ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card");
    write DATASIM_CARD;
    close DATASIM_CARD;

    # Make standard data file links

    system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$* links
    system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

    symlink("$ENV{GTDS_DATA}/sfdir.dat",                    "GTDS\$001");
    symlink("$ENV{GTDS_DATA}/atmosden.dat",                 "GTDS\$002");
    symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",        "GTDS\$008");
    symlink("$ENV{GTDS_DATA}/errormsg.dat",                 "GTDS\$013");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat",  "GTDS\$014");
    symlink("$ENV{GTDS_DATA}/newcomb.dat",                  "GTDS\$023");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat",  "GTDS\$038");
    symlink("$ENV{GTDS_DATA}/jrdat_nomn_new.dat",           "GTDS\$075");
    symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat", "GTDS\$078");

    # Inflate .orbit file
```

```
foreach $fh ("STDOUT","LOGINFO") {
   print $fh ("Inflating .orbit file...\n");
}
system "gunzip -v $ENV{ATM_EPHEM}/${ephem_opt}/${orbit_file}.gz";

# Make job-specific data links

symlink("$ENV{ATM_DATASIM}/${datasim_opt}/$datasim_card","GTDS\$005");
symlink("$ENV{ATM_DATASIM}/${datasim_opt}/$output_file", "GTDS\$006");
symlink("$ENV{ATM_EPHEM}/${ephem_opt}/$orbit_file",     "GTDS\$020");

# Run GTDS!

foreach $fh ("STDOUT","LOGINFO") {
   print $fh "\nUNIX-GTDS\n";
   print $fh "Charles Stark Draper Laboratory\n\n";
   print $fh ("Run started at: ", get_time(), "\n");
}

undef $child_id;

if ($child_id = fork) {         # Parent process here

   local $SIG{USR1} = sub {   # Define anonymous sub to kill GTDS

       (my $gtds_id) = split (" ", `ps | grep gtds`);

       foreach $fh ("STDOUT","LOGINFO") {
          print $fh "GTDS run has exceeded $time_limit seconds;\n";
          print $fh "Killing process $gtds_id\n";
       }

       kill 'QUIT', $gtds_id;
   };
   waitpid $child_id, 0;       # Wait for child process to finish
}

elsif (defined $child_id) {    # Child process here

   $par_id = getppid;

   local $SIG{ALRM} = sub {   # Define local ALRM signal handler
       kill 'USR1', $par_id;  # Send USR1 signal to parent if local alarm goes off
       foreach $fh ("STDOUT","LOGINFO") {
          print $fh "Sending USR1 to $par_id..\n";
       }
   };

   alarm $time_limit;          # Initialize alarm to go off in $time_limit sec
#       system("$ENV{GTDS_EXE}/gtds.exe");
   system("$ENV{GTDS_EXE_DBG}/gtds_dbg.exe");  # Need to run debug version if noise!
   alarm 0;                    # Turn off alarm if GTDS finishes before $time_limit
   die "Exiting child process...";
}

foreach $fh ("STDOUT","LOGINFO") {
   print $fh ("Run ended at: ", get_time(), "\n");
}

system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$* links
system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

foreach $fh ("STDOUT","LOGINFO") {                        # Recompress .orbit file
   print $fh ("Compressing .orbit file...\n");
}
system "gzip -v $ENV{ATM_EPHEM}/${ephem_opt}/$orbit_file";

# Read .output file and create OBSCARD file (FRN 15)

foreach $fh ("STDOUT","LOGINFO") {
   print $fh ("Writing OBSCARD\n");
```

```
    }

    open OUTFILE, "<$ENV{ATM_DATASIM}/${datasim_opt}/$output_file";
    open OBSCARD, ">$ENV{ATM_DATASIM}/${datasim_opt}/$obs_file";
    printf OBSCARD "OBSCARD \n";

  OBSLINE: while (defined($outline = <OUTFILE>)) {
      if ($outline =~ m/
          ^\s{0,1}(\d{6,7})\s+
          (\d{5,6}\.\d{3})\s+
          (\w{4})\s+
          (\w+)\s+
          (0\.\d{16})D([-+]\d{2})
          /x) {
          $ymd = $1;
          $hms = sprintf "%010.3f", $2;
          $statid = $3;
          $type = $obstype{$4};
          $observtn = sprintf("%16.14fE%3s", $5, $6);
          if (($type == 4) || ($type == 5)) {
              $observtn = ($observtn*$PI)/180;  # Convert to radians
          }
          write OBSCARD;
      }
      elsif ($outline =~ /^\s+RETURN 1/) {
          printf OBSCARD "END     \n";
          last OBSLINE;
      }
  }

    close OUTFILE;
    close OBSCARD;

}

close LOGINFO;

#==============   OBSCARD file formatting      ==================================

format OBSCARD =
@<<<    @>>      @>>>>>>@<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
$statid, $type,  $ymd,  $hms,         $observtn,           $observtn
.

#==============   DATASIM card deck formatting ==================================

format DATASIM_CARD =
CONTROL   DATAMGT                                                @<<<<<<<  @>>>>>>>
                                                                 $intl_des, $catnum

OGOPT
POTFIELD  1 4
END
FIN
CONTROL   DATASIM                                                @<<<<<<<  @>>>>>>>
                                                                 $intl_des, $catnum
DMOPT
/FLYQ   1 0346  3   338.900            541242.8299        3591947.6900
/PARQ   1 0396  3   347.300            484329.1839        2620600.8719
/EGLQ   1 0399  3     0.380            303420.7790        2734706.5526
/KAEQ   1 0932  3   300.459648         213419.4537        2014359.7376002
END
DCOPT
DSPEA1    1  0    @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< 60.0
                  $start_ymd,          $start_hms
DSPEA2   20  1  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                  $end_ymd,            $end_hms
DSPEA3    2  1  0
/FLYQ   0 1  4  5    35.0                54.0                54.0
/PARQ   0 1  4  5    48.0                54.0                46.8
/EGLQ   0 1  4  5    30.0                45.0                45.0
/KAEQ   0 1  4  5    4.631               29.5                30.42
```

```
ELLMODEL  1        6378.135            298.26
/FLYQ   200001
/PARQ   200001
/EGLQ   200001
/KAEQ   200001
/FLYQ   7 1   @>>  60.0                 24.0                5.0
              $span_len
/PARQ   7 1   @>>  60.0                 24.0                5.0
              $span_len
/EGLQ   7 1   @>>  60.0                 24.0                5.0
              $span_len
/KAEQ   7 1   @>>  120.0                24.0                5.0
              $span_len
/FLYQ   9 1       @>>>>>>@<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                  $start_ymd, $start_hms, $interval1_ymdhms
/PARQ   9 1       @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                  $interval1_ymdhms,   $interval2_ymdhms
/EGLQ   9 1       @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                  $interval2_ymdhms,   $interval3_ymdhms
/KAEQ   9 1       @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<
                  $interval3_ymdhms,   $interval4_ymdhms
TRACKELV  3         5.0
END
FIN
.
```

C.3 The estbfs.pl Program

```
#
# estbfs.pl - Ballistic Factor Estimator Program
#
# Author:
#
# George R. Granholm
# 09 Apr 00
#
#

# Import modules

use Dates;                      # For cal2jul, jul2cal, & get_time subroutines
use Localmath;                  # For round subroutine
use FileHandle;                 # for autoflush;
sub numerically { $a <=> $b }; # To sort in ascending order

# Set options & variables

$start_epoch = "991215 000000.0";
$end_epoch   = "1000211 000000.0";
$ephem_opt   = "lowgrav";
$datasim_opt = "lowgrav_noise";
$dc_opt      = "lowgrav_schatten_noise";
$initfile    = "$ENV{ATM_CAL}/${dc_opt}/initinfo.txt";
$blfcfile    = "$ENV{ATM_CAL}/${dc_opt}/ballfcts.txt";
$print_sched = 0;               # Flag to print pass schedule; 1 = yes, 0 = no
$increment   = 0.125;           # Shift span for each DC by this much (days)
$fit_len     = 3;               # Length of each fit span (days)
$diverge_tol = 11;              # Allowed length of "sparse" area in data (days)
                                # (or num. of days of consecutive divergent runs)
$time_limit  = 180;               # Allowed duration of GTDS run (secs)
$rho1_tol    = 10;              # Max absolute value of $rho1
$num_procs   = 8;               # Number of processes to spawn (including parent)

# Read and format start epoch

($start_ymd, $start_hms) = split(" ", $start_epoch);
if (length($start_ymd) == 7) {
    ($start_ymd) = ($start_ymd =~ /(\d{6})$/);          # Take off GTDS Y2K fix
```

```
}                                                          # for Julian date conversion
($y,$m,$d) = ($start_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($start_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$start_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Read and format end epoch

($end_ymd, $end_hms) = split(" ", $end_epoch);
if (length($end_ymd) == 7) {
    ($end_ymd) = ($end_ymd =~ /(\d{6})$/);
}
($y,$m,$d) = ($end_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
($h,$mn,$s) = ($end_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
$end_jul = cal2jul($y,$m,$d,$h,$mn,$s);

# Open and read $initfile

open INITINFO, "$initfile" or die "Can't find $initfile";

 INITLINE: while (defined($line = <INITINFO>)) {

     $line =~ s/^(\d{5})\s//;
     $initinfo{$1} = [ split(" ",$line) ];
 }

close INITINFO;
@initinfo = sort keys %initinfo;

# Open output file so that all processes can access it

open BALLFCTS, ">>$blfcfile";

# Spawn appropriate number of processes

$child_id[1] = $$;  # Parent process number

 SPAWN: for ($proc_num = 2; $proc_num <= $num_procs; $proc_num++) {
     $child_id[$proc_num] = fork;       # The parent knows all process nums
     if ($child_id[$proc_num] == 0) {  # The child only knows the parent's
        $child_id[$proc_num] = $$;     # And its own process num
        last SPAWN; }
 }

# Create subdirectories for each process

if ($$ == $child_id[1]) { $proc_num = 1; }
$dc_opt .= "/run${proc_num}";
mkdir "$ENV{ATM_CAL}/${dc_opt}",0777;
mkdir "$ENV{ATM_DC}/${dc_opt}",0777;
chdir "$ENV{ATM_CAL}/${dc_opt}";
$logfile = "$ENV{ATM_CAL}/${dc_opt}/estbfs.log";

# Open or redirect files

open LOGINFO, ">>$logfile";
open STDERR, ">>&LOGINFO";   # Redirect STDERR to LOGINFO

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "BALLFCTS") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\testbfs.pl: Processing ${initfile}\n";
    print $fh "\tProcess \# ${proc_num}\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
```

```
}

# Assign chunk of @initinfo to each process

for ($i = 1; $i <= $num_procs; $i++) {
    $cutoff[$i] = int(($#initinfo/$num_procs)*$i);
}
$cutoff[0] = -1;

@objects = @initinfo[($cutoff[$proc_num-1]+1)..$cutoff[$proc_num]];

# Begin main loop by $catnum

 OBJLOOP: foreach $catnum (@objects) {

     # Initialize variables

     $intl_des = $initinfo{$catnum}[0];

     $ephem_output = "$ENV{ATM_EPHEM}/${ephem_opt}/${catnum}_ephem.output";
     $datasim_output = "$ENV{ATM_DATASIM}/${datasim_opt}/${catnum}_datasim.output";

     # Read in array of observation times

     open SIMOUT, $datasim_output;
     $index = 0;
     while (defined($simline = <SIMOUT>)) {
        if ($simline =~ /^\s+INTERVAL\s{1,2}\d{1,2}/) {
            foreach $i (1..4) {
                $simline = <SIMOUT>;
                if ($simline =~ m{              # Match start of pass
                    ^\s+TIME\s1ST\sOB\s\=\s*
                    (\d+)\s+
                    (\d+)\s+
                    (\d+\.\d+)
                    }x) {
                    $obstart_ymd = $1;
                    $obstart_hms = sprintf("%04d", $2) . sprintf("%06.3f",$3);
                }
                elsif ($simline =~ m{           # Match end of pass
                    ^\s+TIME\sLAST\sOB\=\s*
                    (\d+)\s+
                    (\d+)\s+
                    (\d+\.\d+)
                    }x) {
                    $obend_ymd = $1;
                    $obend_hms = sprintf("%04d", $2) . sprintf("%06.3f",$3);

                    if (length($obstart_ymd) == 7) {
                        ($obstart_ymd) = ($obstart_ymd =~ /(\d{6})$/);
                    }
                    ($y,$m,$d) = ($obstart_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
                    ($h,$mn,$s) = ($obstart_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
                    $obstart_jul = cal2jul($y,$m,$d,$h,$mn,$s);

                    if (length($obend_ymd) == 7) {
                        ($obend_ymd) = ($obend_ymd =~ /(\d{6})$/);
                    }
                    ($y,$m,$d) = ($obend_ymd =~ /^(\d{2})(\d{2})(\d{2})/);
                    ($h,$mn,$s) = ($obend_hms =~ /^(\d{2})(\d{2})(\d{2}[.\s]*\d*)/);
                    $obend_jul = cal2jul($y,$m,$d,$h,$mn,$s);

                    if ($obstart_jul != $obend_jul) {       # If pass contains any obs
                        $obstart[$index] = $obstart_jul;    # Store in respective arrays
                        $obend[$index] = $obend_jul;
                        $index++;
                    }
                }
            }
        }
     }
```

```
    close SIMOUT;

    # Sort and parse array of observation times

    @obstart = sort numerically @obstart;
    @obend = sort numerically @obend;

    for ($i = 1; $i <= $#obstart; $i++) {     # Eliminate overlap
        if ($obstart[$i] <= $obend[$i-1]) {
            splice(@obstart,$i,1);
            splice(@obend,$i-1,1);
            $i -= 1;
        }
    }

    if ($print_sched) {
        open SIMSCHED, ">$ENV{ATM_DC}/${dc_opt}/${catnum}_sched.txt";
        for ($i = 0; $i <= $#obend; $i++) {
            print SIMSCHED "Span ${i}: $obstart[$i] - $obend[$i]\n";
        }
        close SIMSCHED;
    }

    # Calculate mass and area for DC

    $ball_fact = $initinfo{$catnum}[1];   # Can be "perfect" B value or with error
    $Ax = $initinfo{$catnum}[2];
    $C_d = 2.2;                           # Default for LEO
    $mass = ($Ax*$C_d)/(2*$ball_fact);    # in kg
    $Ax_km = sprintf("%7.10E",($Ax/1000000));          # Convert to km^2

#   $var = $initinfo{$catnum}[3];
#   $stan_flag = $initinfo{$catnum}[4];
#   $obs_type = $initinfo{$catnum}[5];

    # Initialize DC start and end epochs

    $dc_start_jul = $start_jul;
    $dc_end_jul = $start_jul + $fit_len;
    $div_cnt = 0;                          # Identifies last run that converged
    $run_num = 1;
    $first_run = 1;
    $in_span = 1;
    $have_obs = 1;
    undef %conv_epoch;                     # Hash of converged epochs

    # Begin loop for DC spans

  DCLOOP: while ($in_span) {

      $converged = 0;
      $i = 0;
      unless ($first_run) {$have_obs = 0};

      # Test if there are any new observations for this object

    TESTOBS: while (!$first_run && ($i <= $#obstart)) {

        if ((($obstart[$i] >= ($dc_end_jul - $increment)) &&
             ($obstart[$i] <= $dc_end_jul)) or
            (($obend[$i] >= ($dc_end_jul - $increment)) &&
             ($obend[$i] <= $dc_end_jul))) {
            $have_obs = 1;
            last TESTOBS;
        }
    } continue {$i++;}

      next DCLOOP unless ($have_obs);

      # Continue with run
```

```
        foreach $fh ("STDOUT","LOGINFO") {
           print $fh "-" x 40,"\n";
           print $fh "  Processing NORAD Catalog \#$catnum\n";
           print $fh "  Process \# $proc_num\n";
           print $fh "  Run number $run_num\n";
        }

        @dc_start_cal = jul2cal($dc_start_jul);

        # Check if $diverge_tol has been exceeded; assign Julian date
        # to look for in .output file and assign epoch & epoch advance date

        if (!$first_run && (($dc_start_jul - (cal2jul(@{ $conv_epoch{$div_cnt} }))))
                          > $diverge_tol)) {
           foreach $fh ("STDOUT","LOGINFO") {
               print $fh "Object $catnum not converged for $diverge_tol consecutive
days;\n";
               print $fh "Going to next object\n";
           }
           next OBJLOOP;
        }
        elsif (!$first_run && (($conv_epoch{$div_cnt}[1] ==
                               $dc_start_cal[1]) &&          # If last epoch that converged
                              ($conv_epoch{$div_cnt}[2] ==
                               $dc_start_cal[2]))) {         # is on same day as curr. epoch
           $read_jul = sprintf("%12.4f",$dc_start_jul);
           @epoch = @dc_start_cal;
           $epoch_adv = 0;
        }

        else {  # Either first run or epoch is on different day as last conv. epoch
           $read_jul = sprintf("%12.4f",cal2jul(@dc_start_cal[0..2],0,0,0));
           @epoch = (@dc_start_cal[0..2],0,0,0);
           if (($dc_start_cal[3] == 0) && ($dc_start_cal[4] == 0) &&
               ($dc_start_cal[5] == 0)) {$epoch_adv = 0;}
           else {
               $epoch_adv = 1;
               @epoch_adv = @dc_start_cal;
           }
        }

        # Format $epoch_adv for GTDS

        if ($epoch_adv) {
           ($epoch_adv[0]) = ($epoch_adv[0] =~ /\d{2}(\d{2})$/);
           if ($epoch_adv[0] == 0) {$epoch_adv[0] = 100};
           $epoch_adv_ymd = join("",@epoch_adv[0..2]);
           $epoch_adv_hms = join("",@epoch_adv[3..5]);
        }
        else {
           $epoch_adv_ymd = "";
           $epoch_adv_hms = "";
        }

        # Format rest of dates for GTDS

        $dc_strt_eph_jul = cal2jul(@dc_start_cal[0..2],0,0,0) + 1; # Beg of day aftr epch

        @dc_end_cal      = jul2cal($dc_end_jul);
        @dc_strt_eph_cal = jul2cal($dc_strt_eph_jul);
        @dc_end_eph_cal  = jul2cal($dc_start_jul + $diverge_tol);  # Allowd num w/o convrg

        ($epoch[0])           = ($epoch[0] =~ /\d{2}(\d{2})$/);
        ($dc_start_cal[0])    = ($dc_start_cal[0] =~ /\d{2}(\d{2})$/);
        ($dc_end_cal[0])      = ($dc_end_cal[0] =~ /\d{2}(\d{2})$/);
        ($dc_strt_eph_cal[0]) = ($dc_strt_eph_cal[0] =~ /\d{2}(\d{2})$/);
        ($dc_end_eph_cal[0])  = ($dc_end_eph_cal[0] =~ /\d{2}(\d{2})$/);

        if ($epoch[0] == 0)           {$epoch[0] = "100";}
        if ($dc_start_cal[0] == 0)    {$dc_start_cal[0] = "100";}
        if ($dc_end_cal[0] == 0)      {$dc_end_cal[0] = "100";}
```

```
if ($dc_strt_eph_cal[0] == 0) {$dc_strt_eph_cal[0] = "100";}
if ($dc_end_eph_cal[0] == 0)  {$dc_end_eph_cal[0] = "100";}

$epoch_ymd       = join("",@epoch[0..2]);
$epoch_hms       = join("",@epoch[3..5]);
$dc_start_ymd    = join("",@dc_start_cal[0..2]);
$dc_start_hms    = join("",@dc_start_cal[3..5]);
$dc_end_ymd      = join("",@dc_end_cal[0..2]);
$dc_end_hms      = join("",@dc_end_cal[3..5]);
$dc_strt_eph_ymd = join("",@dc_strt_eph_cal[0..2]);
$dc_strt_eph_hms = "000000.0";
$dc_end_eph_ymd  = join("",@dc_end_eph_cal[0..2]);
$dc_end_eph_hms  = "000000.0";

# Assign input and output file names

if ($first_run) {$dc_input_file = $ephem_output;}
else {$dc_input_file =
        "$ENV{ATM_DC}/${dc_opt}/${catnum}_dc_${div_cnt}.output";}

# Get a-priori elements from appropriate .output file

open INFILE, $dc_input_file;
$endflag = 0;

if ($first_run) {   # Then read from EPHEM .output file

 EPHEMLINE: while (defined($inline = <INFILE>)) {

     if ($inline =~ /^ ENTERED ORBINT/) {
        $endflag = 1;
     }
     elsif ($endflag && ($inline =~ /^ DATE.*JULIAN DATE = $read_jul/ )) {
        while (defined($inline = <INFILE>)) {
            if ($inline =~ m{
                ^\sX\s*(-*\d+\.\d+)
                \s*Y\s*(-*\d+\.\d+)
                \s*Z\s*(-*\d+\.\d+)
                \s*DX\s*(-*\d+\.\d+)
                \s*DY\s*(-*\d+\.\d+)
                 \s*DZ\s*(-*\d+\.\d+)
                }x ) {
                @aprioris = ($1,$2,$3,$4,$5,$6);
                last EPHEMLINE;
            }
        }
     }
 }
}

else {              # Read from appropriate DC .output file

 DCLINE: while (defined($inline = <INFILE>)) {

     if ($inline =~ /^ DATE.*JULIAN DATE = $read_jul/ ) {
        while (defined($inline = <INFILE>)) {
            if ($inline =~ m{
                ^\sX\s*(-*\d+\.\d+)
                \s*Y\s*(-*\d+\.\d+)
                \s*Z\s*(-*\d+\.\d+)
                \s*DX\s*(-*\d+\.\d+)
                \s*DY\s*(-*\d+\.\d+)
                 \s*DZ\s*(-*\d+\.\d+)
                }x ) {
                @aprioris = ($1,$2,$3,$4,$5,$6);
                last DCLINE;
            }
        }
     }
 }
}
```

```
close INFILE;

# Write GTDS DC card file

$dc_card = "${catnum}_dc_${run_num}.gtds";
$obs_file = "${catnum}_datasim.obscard";
$dc_output_file = "${catnum}_dc_${run_num}.output";

open DC_CARD, ">$ENV{ATM_DC}/${dc_opt}/$dc_card";
write DC_CARD;
close DC_CARD;

# Make standard data file links

system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$* lnks
system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files

symlink("$ENV{GTDS_DATA}/sfdir.dat",                    "GTDS\$001");
symlink("$ENV{GTDS_DATA}/atmosden.dat",                 "GTDS\$002");
symlink("$ENV{GTDS_DATA}/radarsat_earthfld.dat",        "GTDS\$008");
symlink("$ENV{GTDS_DATA}/errormsg.dat",                 "GTDS\$013");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.mn1950.dat",  "GTDS\$014");
symlink("$ENV{GTDS_DATA}/newcomb.dat",                  "GTDS\$023");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.timcof.dat",  "GTDS\$038");
symlink("$ENV{GTDS_DATA}/jrdat_nomn.dat",               "GTDS\$075");
symlink("$ENV{GTDS_DATA}/june94.msgen.slp.tod1950.dat", "GTDS\$078");

# Make job-specific data links

symlink("$ENV{ATM_DC}/${dc_opt}/$dc_card",          "GTDS\$005");
symlink("$ENV{ATM_DC}/${dc_opt}/$dc_output_file",   "GTDS\$006");
symlink("$ENV{ATM_DATASIM}/${datasim_opt}/$obs_file",   "GTDS\$015");

# Run GTDS!

foreach $fh ("STDOUT","LOGINFO") {
    print $fh "  Epoch ${dc_start_ymd} ${dc_start_hms}\n";
    print $fh "-" x 40,"\n";
    print $fh "\nUNIX-GTDS\n";
    print $fh "Charles Stark Draper Laboratory\n\n";
    print $fh ("Run started at: ", get_time(), "\n");
}

undef $grandchild_id;

if ($grandchild_id = fork) {        # Parent or first-generation child process

    local $SIG{USR1} = sub {        # Define anonymous sub to kill GTDS

        $ps = `ps -f | grep $grandchild_id | grep gtds`;
        ($uid,$gtds_id) = split (" ",$ps);
        foreach $fh ("STDOUT","LOGINFO") {
            print $fh "GTDS run has exceeded time limit;\n";
            print $fh "Killing process $gtds_id\n";
        }
        kill 'QUIT', $gtds_id;

    };
    waitpid $grandchild_id, 0;      # Wait for child process to finish
}

elsif (defined $grandchild_id) {    # Grandchild process

    local $SIG{ALRM} = sub {   # Define local ALRM signal handler
        kill 'USR1', $child_id[$proc_num];  # Send USR1 signal to parent
                                            # if local alarm goes off
        foreach $fh ("STDOUT","LOGINFO") {
            print $fh "Sending USR1 to $child_id[$proc_num]..\n";
        }
    };
```

```
            alarm $time_limit;          # Initialize alarm to go off in $time_limit sec
            system("$ENV{GTDS_EXE}/gtds.exe");
            alarm 0;                    # Turn off alarm if GTDS finishes before $time_limit
            die "Exiting grandchild process...\n";
        }

        foreach $fh ("STDOUT","LOGINFO") {
            print $fh ("Run ended at: ", get_time(), "\n");
        }

        # Test if run converged; set flags and read $rho1, $ht_per

        open OUTFILE, "GTDS\$006";
        $ht_per = 0;
        $rho1 = 0;

      OUTLINE: while (defined($outline = <OUTFILE>)) {
            if (!$converged && ($outline =~ /^\s+\*{5} DC CONVERGED/)) {
                $converged = 1;
                $div_cnt = $run_num;
                $conv_epoch{$run_num} = [ @dc_start_cal ];
                if ($conv_epoch{$run_num}[0] > 99) {  # Remove GTDS formatting if necessary
                    $conv_epoch{$run_num}[0] -= 100;
                    $conv_epoch{$run_num}[0] = sprintf("%02d", $conv_epoch{$run_num}[0]);
                }
                if ($first_run) {$first_run = 0;}
            }
            if ($converged) {
                if (!$ht_per && ($outline =~ /^\s+HT\. OF PERIFOCUS\s+(\d+\.\d+)\s/)) {
                    $ht_per = $1;
                }
                if ($ht_per && ($outline =~ s/^\s+AERO VARIATION \(RHO1\)\s+=\s*(-
*\d\.\d{8})D([+-]\d{2})/$1e$2/)) {
                    $rho1 = $outline;

                    # Throw out $rho1 values that are obviously not valid

                    if (abs($rho1) > $rho1_tol) {$converged = 0;}
                    last OUTLINE;
                }
            }
        }

        close OUTFILE;

        # If converged, write to log, write line to ballfcts.txt

        if ($converged) {
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh ("Run converged\n");
            }
            $attrib_time = ($dc_start_jul + $dc_end_jul)/2;
            $C_d_est = $C_d*(1+$rho1);
            $B_est = ($C_d_est*$Ax)/(2*$mass);
            printf BALLFCTS "%5s %12.4f %7.10E %7.10E\n", $catnum, $attrib_time, $B_est,
$ht_per;
        }

        else {
            foreach $fh ("STDOUT","LOGINFO") {
                print $fh ("Run diverged or bad rho1: $rho1\n");
            }
        }

        system q { /usr/bin/tcsh -c 'rm GTDS\$* >& /dev/null' };  # Remove any GTDS$*
links
        system q { /usr/bin/tcsh -c 'rm tmp.* >& /dev/null' };    # Remove any temp files
        system q { /usr/bin/tcsh -c 'rm core >& /dev/null' };     # Remove core

    } continue {
```

```
      $dc_start_jul += $increment;
      $dc_end_jul += $increment;
      $run_num += 1;
      if ($dc_end_jul > $end_jul) {$in_span = 0;}
   }

    foreach $fh ("STDOUT","LOGINFO") {
       print $fh ("Compacting .output and .gtds files...\n");
    }

 } continue {
    system qq! tar cf $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_all.output.tar \\
                  $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_\[0-9\]\*.output;
               gzip -v $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_all.output.tar;
               rm $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_\[0-9\]\*.output; !;
    system qq! tar cf $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_all.gtds.tar \\
                  $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_\[0-9\]\*.gtds;
               gzip -v $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_all.gtds.tar;
               rm $ENV{ATM_DC}/${dc_opt}/${catnum}_dc_\[0-9\]\*.gtds; !;
 }

close LOGINFO;

# If parent, wait for slow-finishing children processes

if ($proc_num == 1) {
    for ($i = 2; $i <= $num_procs; $i++) {
       waitpid $child_id[$i],0;
    }
    close BALLFCTS;
}

#==============   DC card deck formatting   ====================================

format DC_CARD =
CONTROL   DC                                                  @<<<<<<<  @>>>>>>>
                                                              $intl_des, $catnum
EPOCH               @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @>>>>>>@<<<<<<<<<<<
                    $epoch_ymd,         $epoch_hms, $epoch_adv_ymd, $epoch_adv_hms
ELEMENT1  1  1  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $aprioris[0],       $aprioris[1],       $aprioris[2]
ELEMENT2            @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                    $aprioris[3],       $aprioris[4],       $aprioris[5]
ORBTYPE   2  1  1 60.
OBSINPUT  5         @>>>>>>@<<<<<<<<<<<< @>>>>>>@<<<<<<<<<<<
             $dc_start_ymd, $dc_start_hms, $dc_end_ymd, $dc_end_hms
DMOPT
/FLYQ    1 0346  3    338.900              541242.8299         3591947.6900
/PARQ    1 0396  3    347.300              484329.1839         2620600.8719
/EGLQ    1 0399  3      0.380              303420.7790         2734706.5526
/KAEQ    1 0932  3    300.459648           213419.4537         2014359.7376002
END
DCOPT
/FLYQ    0 1   4  5     35.0                 54.0                54.0
/PARQ    0 1   4  5     48.0                 54.0                46.8
/EGLQ    0 1   4  5     30.0                 45.0                45.0
/KAEQ    0 1   4  5     4.631                29.5                30.42
ELLMODEL  1          6378.135             298.26
/FLYQ    200001
/PARQ    200001
/EGLQ    200001
/KAEQ    200001
TRACKELV  3          5.0
EDIT                 3.0
PRINTOUT  1      1
CONVERG  25  6       1.0D-4
END
OGOPT
DRAG      1          1
ATMOSDEN          1
```

```
DRAGPAR   3  0    @<<<<<<<<<<<<<<<<<<<
                  $C_d
DRAGPAR   1
SCPARAM           @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                  $Ax_km,             $mass
MAXDEGEQ  1        4
MAXORDEQ  1        4
MAXDEGVE  1        4
MAXORDVE  1        4
POTFIELD  1  4
SOLRAD    1        1.0
END
FIN
CONTROL   EPHEM                       OUTPUT              @<<<<<<<  @>>>>>>>
                                                          $intl_des, $catnum
OUTPUT    1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<  10800.0
                  $dc_strt_eph_ymd, $dc_strt_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN       1
DRAG      1        1
DRAGPAR   3  0    2.2
SCPARAM           @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                  $Ax_km,             $mass
POTFIELD  1  4
MAXDEGEQ  1        4.0
MAXORDEQ  1        4.0
SOLRAD    1        1.0
END
FIN
CONTROL   EPHEM                       OUTPUT              @<<<<<<<  @>>>>>>>
                                                          $intl_des, $catnum
OUTPUT    1  2  1 @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<  86400.0
                  $dc_end_eph_ymd, $dc_end_eph_hms
ORBTYPE   2  1  1  60.0
OGOPT
ATMOSDEN       1
DRAG      1        1
DRAGPAR   3  0    2.2
SCPARAM           @<<<<<<<<<<<<<<<<<<< @<<<<<<<<<<<<<<<<<<<
                  $Ax_km,             $mass
POTFIELD  1  4
MAXDEGEQ  1        4.0
MAXORDEQ  1        4.0
SOLRAD    1        1.0
END
FIN
.
```

C.4 The calcvars.pl Program

```
#
# calcvars.pl - Density Variation Calculator Program
#
# Author:
#
# George R. Granholm
# 30 Apr 00
#
#

# Import modules

use Dates;                    # For cal2jul, jul2cal, & get_time subroutines
use Localmath;                # For round subroutine
use FileHandle;               # for autoflush;
sub numerically { $a <=> $b }; # To sort in ascending order
```

```
# Set options & variables

$model_opt = "lowgrav_schatten_noise";
$logfile   = "$ENV{ATM_CAL}/${model_opt}/calcvars.log";
$initfile  = "$ENV{ATM_CAL}/${model_opt}/initinfo.txt";
$blfcfile  = "$ENV{ATM_CAL}/ballfcts.txt";
$sortdfile = "$ENV{ATM_CAL}/${model_opt}/ballfcts_sort.txt";
$tmpfile   = "$ENV{ATM_CAL}/${model_opt}/array_tmp.txt";
$tau_min   = .125;     # Minimum length of each span j (days)
$min_num_k = 35;       # Minimum number of ballistic factor estimation per span j
$increment = .125;     # Increment to add to $tau_min

# Define f_1 and f_2 (linear density variation functions)

sub f_1 {
    return "1";
}

sub f_2 {
    my $h = shift(@_);
    my $value = ($h - 400)/200;
    return $value;
}

# Open or redirect files

open LOGINFO, ">>$logfile";
open STDERR, ">>&LOGINFO";   # Redirect STDERR to LOGINFO
open TMPFILE, ">$tmpfile";

foreach $fh ("STDOUT", "LOGINFO", "STDERR", "TMPFILE") {
    $fh->autoflush(1);
}

# Write header to $logfile and STDOUT

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh "-" x 50,"\n";
    print $fh "\tcalcvars.pl: Processing ${initfile}\n";
    print $fh "-" x 50,"\n";
    print $fh ("\tJob started at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

# Open and read $initfile

open INITINFO, "$initfile" or die "Can't find $initfile";

 INITLINE: while (defined($line = <INITINFO>)) {

     $line =~ s/^(\d{5})\s//;
     $initinfo{$1} = [ split(" ",$line) ];
 }

close INITINFO;

# Sort $blfcfile by attribution time

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "Sorting ballistic factors by attribution time...\n";
}

system "sort -nk2,2 $blfcfile > $sortdfile";

# Read $sortdfile into @blfcs array

undef $line;
open BLFCFILE, "$sortdfile" or die "Can't find $sortdfile";
$index = 0;
```

```
while (defined($line = <BLFCFILE>)) {
    $blfcs[$index] = [ split(" ",$line) ];
    $index++;
}

close BLFCFILE;

$j = 0;
$span_time[0] = $blfcs[0][1];
$end_time = $blfcs[$#blfcs][1];
$i_save = 0;

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "Building $tmpfile...\n";
}

# Begin main loop

while ($span_time[$j] <= $end_time) {

    $tau[$j] = $tau_min;

  COUNTBLFCS: for ($i = $i_save; $i <= $#blfcs; $i++) {
      if ($blfcs[$i][1] < ($span_time[$j] + $tau[$j])) {
          # Put in test for negative ball. factors here??
          $temp_array[$i-$i_save] = $blfcs[$i];
      }
      else {
          $i_save = $i;
          last COUNTBLFCS;
      }
  }

    # Test for enough estimations in span

    if ($#temp_array < $min_num_k) {
        $tau[$j] += $increment;
        $i_save  -= ($#temp_array+1);
        goto COUNTBLFCS;
    }

    else {

        # Define arrays for MATLAB input

        foreach $fh ("STDOUT", "LOGINFO") {
            print $fh "Span $j: [$span_time[$j],",
            $span_time[$j] + $tau[$j],")\n";
        }

        undef @F;
        undef @a;
        undef @P;

        print TMPFILE "$span_time[$j] ", ($#temp_array+1), "\n";

        for ($n = 0; $n <= $#temp_array; $n++) {
            $F[$n] = [ f_1($temp_array[$n][3]), f_2($temp_array[$n][3]) ];
            $a[$n] = ($temp_array[$n][2]/$initinfo{$temp_array[$n][0]}[1]) - 1;
            $P[$n] = 1/$initinfo{$temp_array[$n][0]}[3];
            print TMPFILE "$F[$n][0] $F[$n][1] $a[$n] $P[$n]\n";
        }
    }

} continue {
    undef @temp_array;
    $j++;
    $span_time[$j] = $span_time[$j-1] + $tau[$j-1];
}

close TMPFILE;
```

```
# End program

foreach $fh ("STDOUT", "LOGINFO") {
    print $fh "-" x 50,"\n";
    print $fh ("\tJob ended at ", get_time(), "\n");
    print $fh "-" x 50,"\n";
}

close LOGINFO;
```

C.5 The calc_b.m Program

```
%
% Author:
%
% George R. Granholm
% 1 May 00
%
%

clear all;
warning off;
more off;

% Set environment variables and filenames

[status,ATM_CAL] = unix('echo $ATM_CAL');
[status,ATM_DC]  = unix('echo $ATM_DC');
ATM_CAL = ATM_CAL(1:length(ATM_CAL)-1);  % Remove newline
ATM_DC  = ATM_DC(1:length(ATM_DC)-1);    % Remove newline

model_opt = 'lowgrav_schatten_noise';
tmpfile   = 'array_tmp.txt';
outfile   = 'jac_densvars_sn_25d.txt';
logfile   = 'calc_b.log';

% Open files and initialize variables

logid  = fopen(strcat(ATM_CAL,'/',model_opt,'/',logfile),'a');
toler      = 3;                  % 3 sigma tolerance
frcst_days = 30;                 % Number of days to forecast
T          = 27;                 % Assumed period of density variations
lambda     = 2*pi/T;
time_grid  = .125;               % Time grid for forecasting (days)
sigma_b1   = 0.07;               % Std dev of WGN in b1
sigma_b2   = 0.07;               % Std dev of WGN in b2
sigma_b1_r = 0.4;                % Std dev of Gauss-Markov RP for b1
sigma_b2_r = 0.3;                % Std dev of Gauss-Markov RP for b2
alpha      = 0.241;              % Rate of decay of correlation
calc_flag  = 1;                  % Input flag
                                 %    1 = calculate dens vars
                                 %    2 = read from infile

if (calc_flag==1)

% Begin loop to calculate density variations in data span

tempid = fopen(strcat(ATM_CAL,'/',model_opt,'/',tmpfile),'r');
outid  = fopen(strcat(ATM_CAL,'/',model_opt,'/',outfile),'a');
j = 1;
line  = fgetl(tempid);      % Get first line

while line ~= -1

   clear F a P Pvec;

   values = str2num(line);
```

```
   if length(values) ~= 2
      fprintf(logid,'Error - improper formatting of array_tmp.txt\n');
      disp('Error - improper formatting of array_tmp.txt\n');
      return
   end
   start_time(j) = values(1);
   array_len  = values(2);

% Read in data for span j

   for i = 1:array_len,
      values = str2num(fgetl(tempid));
      F(i,1) = values(1);
      F(i,2) = values(2);
      a(i) = values(3);
      Pvec(i) = values(4);
   end

   P = diag(Pvec);

% Test for erroneous measurements

   a_avg = mean(a);
   a_sigma = std(a);
%   disp(sprintf('a_avg = %7.10e, a_sigma = %7.10e' ,a_avg ,a_sigma));

   delete_count = 0;
   i = 1;

   while i<=array_len,
     if (abs(a(i)-a_avg)/a_sigma) > toler

         F(i,:) = [];      % Delete offending row
         a(i)   = [];      % from matrices or
         P(i,:) = [];      % vectors
         P(:,i) = [];
         array_len = array_len - 1;
         delete_count = delete_count + 1;

      end
      i = i+1;

   end

   disp(sprintf('%3d meas. > %2d-sigma tol.',...
               delete_count, toler));
   fprintf(logid,'%3d meas. > %2d-sigma tol. ',...
               delete_count, toler);

% Calculate b1 and b2 for span j

   b = (inv(F'*P*F))*(F'*P*a');
   densvars(j,1:2) = b';

% Print line to output file

   fprintf(outid,'%12.4f % 10.10E % 10.10E \n',start_time(j),b);
   fprintf(logid,'%12.4f % 10.10E % 10.10E \n',start_time(j),b);
   disp(sprintf('%12.4f % 10.10E % 10.10E',start_time(j),b));

   line = fgetl(tempid);
   j = j + 1;

end

% End loop to calculate density variations in data span

else

% Read dens vars in data span from file
```

```
   outid  = fopen(strcat(ATM_CAL,'/',model_opt,'/',outfile),'a+');
   line   = fgetl(outid);          % Get first line
   j=1;

   while line ~= -1

      values = str2num(line);
      start_time(j) = values(1);
      densvars(j,1:2) = [values(2) values(3)];
      line  = fgetl(outid);
      j = j + 1;

   end

end

% Do forecasting if desired

if (frcst_days)

   fprintf(logid,'Calculating deterministic component...\n');
   disp(sprintf('Calculating deterministic component...'));

   % First solve for deterministic component

   j_max = j - 1;
   t_0 = start_time(j_max);
   for j = 1:j_max,
      G(j,1:3) = [ (1-cos(lambda*(start_time(j) - t_0))) ...
                   cos(lambda*(start_time(j) - t_0))     ...
                   sin(lambda*(start_time(j) - t_0)) ];

   end

   Z_b1 = densvars(:,1);
   Z_b2 = densvars(:,2);

   S_b1 = (inv(G'*G))*(G'*Z_b1);
   S_b2 = (inv(G'*G))*(G'*Z_b2);

   x_bar_b1  = S_b1(1);
   x_bar_b2  = S_b2(1);
   x_0_b1    = S_b1(2);
   x_0_b2    = S_b2(2);
   xdot_0_b1 = S_b1(3);
   xdot_0_b2 = S_b2(3);

   % Calculate estimate of deterministic component over entire time interval

   j_frcst_max = j_max + frcst_days/time_grid;

   for j=1:j_frcst_max,
      if (j>j_max)
         start_time(j) = start_time(j-1) + time_grid;
      end

      determ(j,1) = x_bar_b1 + (x_0_b1-x_bar_b1)*cos(lambda*(start_time(j) - t_0)) ...
                    +(xdot_0_b1/lambda)*sin(lambda*(start_time(j) - t_0));
      determ(j,2) = x_bar_b2 + (x_0_b2-x_bar_b2)*cos(lambda*(start_time(j) - t_0)) ...
                    +(xdot_0_b2/lambda)*sin(lambda*(start_time(j) - t_0));

   end

   % Calculate estimate of random component using scalar Kalman filter

   fprintf(logid,'Calculating random component...\n');
   disp(sprintf('Calculating random component...'));

   p_pred_b1(1) = sigma_b1^2;   % The b1 filter variance at j=1
   p_pred_b2(1) = sigma_b2^2;   % The b2 filter variance at j=1
   x_pred_b1(1) = 0;            % The prediction of b1 at j=1
```

```
  x_pred_b2(1) = 0;              % The prediction of b2 at j=1

  for j=1:j_max,

  % Calculate residuals (which function as measurements of y(j))

     y_b1(j) = densvars(j,1) - determ(j,1);
     y_b2(j) = densvars(j,2) - determ(j,2);

     % Compute Kalman gain

     g_b1(j) = p_pred_b1(j)/(p_pred_b1(j) + sigma_b1^2);
     g_b2(j) = p_pred_b2(j)/(p_pred_b2(j) + sigma_b2^2);

     % Update states and errors based on actual measurement

     x_curr_b1(j) = x_pred_b1(j) + g_b1(j)*(y_b1(j)-x_pred_b1(j));
     x_curr_b2(j) = x_pred_b2(j) + g_b2(j)*(y_b2(j)-x_pred_b2(j));
     p_curr_b1(j) = (p_pred_b1(j)*sigma_b1^2)/(p_pred_b1(j)+sigma_b1^2);
     p_curr_b2(j) = (p_pred_b2(j)*sigma_b2^2)/(p_pred_b2(j)+sigma_b2^2);

     % Prediction ahead to next time step

     tau  = start_time(j+1) - start_time(j);
     x_pred_b1(j+1) = exp(-alpha*tau)*x_curr_b1(j);
     x_pred_b2(j+1) = exp(-alpha*tau)*x_curr_b2(j);
     p_pred_b1(j+1) = exp(-2*alpha*tau)*p_curr_b1(j) + ...
                 (1-exp(-2*alpha*tau))*sigma_b1_r^2;
     p_pred_b2(j+1) = exp(-2*alpha*tau)*p_curr_b2(j) + ...
                 (1-exp(-2*alpha*tau))*sigma_b2_r^2;

  end

  % Save estimates of random component at beginning of forecast span

  x_r_0_b1 = x_curr_b1(j);
  x_r_0_b2 = x_curr_b2(j);

  % Write predicted density variations with deterministic + random components

  for j=j_max+1:j_frcst_max,

     densvars(j,1) = determ(j,1) + exp(-alpha*(start_time(j)-t_0))*x_r_0_b1;
     densvars(j,2) = determ(j,2) + exp(-alpha*(start_time(j)-t_0))*x_r_0_b2;

     fprintf(outid,'%12.4f % 10.10E % 10.10E \n',start_time(j),densvars(j,1:2));
     fprintf(logid,'%12.4f % 10.10E % 10.10E \n',start_time(j),densvars(j,1:2));
     disp(sprintf('%12.4f % 10.10E % 10.10E',start_time(j),densvars(j,1:2)));

  end

end

warning on;

if (calc_flag==1)
   fclose(tempid);
end

fclose(outid);
fclose(logid);
```

[This Page Intentionally Left Blank]

References

[1] Nazarenko, Andrey. "Atmospheric Density Tracking Studies." CSDL-C-6505, Report prepared by the Scientific-Industrial Firm "NUCLON" for the Charles Stark Draper Laboratory, August 1999.

[2] Battin, R.H. *An Introduction to the Mathematics and Methods of Astrodynamics.* AIAA Educational Series. New York: AIAA, Inc., 1987.

[3] Marcos, F.A. "Accuracy of Atmosphere Drag Models at Low Satellite Altitudes." *Adv. Space Research*, Vol. 10, No 3, 1990, pp. 417-422.

[4] Pardini, C. and L. Anselmo. "Calibration of Semi-Empirical Atmosphere Models Through the Orbital Decay of Spherical Satellites." Paper AAS 99-384, presented at AIAA/AAS Astrodynamics Specialist Conference, August 1999.

[5] *Goddard Trajectory Determination System (GTDS) Mathematical Theory.* NASA Operational GTDS Mathematical Specification. Rev. 1. Ed. Computer Sciences Corporation and NASA Goddard Space Flight Center. Contract NAS 5-31500, Task 213, July 1989.

[6] Marcos, F.A., M.J. Kendra, J.M. Griffin, J. Bass, J. Liu, and D. Larsen. "Precision Low Earth Orbit Determination Using Atmospheric Density Calibration." *Advances in the Astronautical Sciences*, Vol. 97(1), 1998, pp. 515-527.

[7] Viereck, Rodney and J. Joselyn. "Solar Cycle Effects on Thermospheric Density and Satellite Drag." Paper AAS 99-379, presented at AIAA/AAS Astrodynamics Specialist Conference, August 1999.

[8] Marcos, F.A. and J.O. Wise. "Satellite Drag Accuracy Improvements From Neutral Density Model Calibration." Paper AAS 99-381, presented at AIAA/AAS Astrodynamics Specialist Conference, August 1999.

[9] Karr, G.R. "Environmental Dynamics at Orbital Altitudes." NASA CR-2765. Washington, D.C.: NASA, 1976.

[10] Gilbreath, G.C., P.W. Schumacher, M. Davis, E. Lydick, and J. Anderson. "Calibrating the Naval Space Surveillance Fence Using Satellite Laser Ranging." *Advances in the Astronautical Sciences*, 97(1), 1998, pp. 403-416.

[11] Storz, Mark. "Satellite Drag Accuracy Improvements Estimated from Orbital Energy Dissipation Rates." Paper AAS 99-385, presented at AIAA/AAS Astrodynamics Specialist Conference, August 1999.

[12] Cefola, P.J. and A.I. Nazarenko. "Neutral Atmosphere Density Monitoring Based on Space Surveillance System Orbital Data." Presented at AIAA/AAS Astrodynamics Specialist Conference, August 1999.

[13] Cefola, P.J. and A.I. Nazarenko. "Refinement of Satellite Ballistic Factors for the Estimation of Atmosphere Density Variations and Improved LEO Orbit Prediction." Paper AAS 99-203, Presented at AAS/AIAA Space Flight Mechanics Meeting, Feb. 1999.

[14] Nazarenko, A.I. "A-priori and A-posteriori Orbit Prediction Errors Evaluation of Low Height Artificial Earth Satellites." *Cosmic Research*, Vol. 29, No. 4, 1991.

[15] Jacchia, Luigi. "Revised Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles." Smithsonian Astrophysical Observatory Special Report 332. Cambridge: Smithsonian Institution, 1971.

[16] Roberts, Charles E. "An Analytic Model for Upper Atmosphere Densities Based Upon Jacchia's 1970 Models." *Celestial Mechanics*, Vol. 4, Dec 1971, pp. 368-377.

[17] Jacchia, Luigi. "Thermospheric Temperature, Density, and Composition: New Models." Smithsonian Astrophysical Observatory Special Report 375. Cambridge: Smithsonian Inst., 1977.

[18] "Dictionary of Technical Terms for Aerospace Use." Ed. Daniel R. Glover, Jr. http://roland.grc.nasa.gov/~dglover/dictionary/. NASA Lewis Research Center. Accessed Aug 27, 1998.

[19] Barlier, F., C. Jaeck-Berger, J.L., Falin, G. Kockarts and G. Thuillier. "A Thermospheric Model Based on Satellite Drag Data." *Ann. Geophys.*, Vol 34, 1978, pp. 9-24.

[20] Köhnlein, W., D. Krankowsky, P. Lämmerzahl, W. Joos and H. Volland. "A Thermospheric Model of the Annual Variations of He, N, O, N_2, and Ar from the Aeros – NIMS Data." *Journal of Geophysical Research.*, Vol. 74, 1979, pp. 4355-4362.

[21] Alcaydé, D. "An Analytic Static Model of Temperature and Composition from 20 to 2000 km Altitude." *Annale de Géophysique*, Vol. 37-4, 1981, pp. 515-528.

[22] Hedin, A.E. "A Revised Thermospheric Model Based on Mass Spectrometer and Incoherent Scatter Data: MSIS-83." *Journal of Geophysical Research*, Vol. 88, No. A12, 1983, pp.10170-10188.

[23] Hedin, A.E. "MSIS-86 Thermospheric Model." *Journal of Geophysical Research*, Vol. 92, 1987, pp. 4649-4662.

[24] Hedin, A.E. "Extension of the MSIS Thermosphere into the Middle and Lower Atmosphere." *Journal of Geophysical Research.*, Vol. 96, No. A2, Feb. 1991, pp. 1159-1172.

[25] Hickey, M.P. "The NASA Marshall Engineering Thermosphere Model." NASA CR-179359, July 1988.

[26] Sehnal, L. and L. Pospisolova. "Thermospheric Model TD 88." Preprint No. 67, Astronomical Institute, Czechoslovak Academy of Sciences, 1988.

[27] Fuller-Rowell, T.J., D. Rees, S. Quegan, R.J. Moffett, M.V. Codrescu, and G.H. Millward. "A Coupled Thermosphere Ionosphere Model (CTIM)." *STEP Handbook of Ionospheric Models*. Ed. R.W. Schunk, 1996.

[28] Hicks, J.R. *An Adaptive Thermospheric Model to Improve the Prediction of Satellite Orbits*. Master of Science Thesis, The George Washington University, Washington, D.C., April 1997.

[29] Owens, J.K. "NASA Marshall Engineering Thermosphere Model – 1999 Version (MET-99)." NASA Technical Memorandum, NASA Marshall Space Flight Center, Huntsville, Alabama, 1999.

[30] Keating, G.M., J.C. Leary, B.D. Green, O.M. Uy, R.C. Benson, R.E. Erlandson, T.E. Phillips, J.C. Lesho and M.T. Boies. "Neutral and Ion Drag Effects Near the Exobase: MSX Satellite Measurements of He and O^+." *Advances in the Astronautical Sciences*, Vol. 97, 1998, pp. 549-556.

[31] Kockarts, Gaston. "Definition of Space Aeronomy." http://www.oma.be/BIRA-IASB/Scientific/Home.html. Accessed Aug. 27, 1999.

[32] Marcos, F.A. "Requirements for Improved Thermospheric Neutral Density Models." Paper AAS 85-312, Presented at AAS/AIAA Astrodynamics Specialist Conference, Vail, Co, August 1985.

[33] Gaposchkin, E.M. and A.J. Coster. "Evaluation of Recent Atmospheric Density Models." Paper AAS 87-557, Presented at AAS/AIAA Astrodynamics Conference, Kalispell, MT, Aug 1987.

[34] Barker, W.N., T.J. Eller, and L. E. Herder. "A New Approach in Treating the Ballistic Coefficient in the Differential Correction Fitting Program." *Advances in the Astronautical Sciences*, Vol 71, Pt. I. San Diego: Univelt, Inc., 1989.

[35] Wright, J.R. Personal Correspondence with J. Fischer, AF/Draper Fellow, 1997.

[36] Jaeck-Berger, C. and F. Barlier. "Review of Drag Effects on Satellite Orbits for Geodynamic Studies." The Use of Artificial Satellites for Geodesy and Geodynamics, *Proceedings of the International Symposium on Geodesy and Geodynamics*, Athens, Greece, May 1973. National Technical University of Athens: 1974, pp. 275-311.

[37] Fischer, J.D. *The Evolution of Highly Eccentric Orbits*. CSDL-T-1310, Master of Science Thesis, Massachusetts Institute of Technology. Cambridge, MA, June 1998.

[38] Metzinger, R.W. *Validation of the Workstation Version of R&D GTDS*. Charles Stark Draper Laboratory: February, 1993. Copy available through Dr. Paul Cefola, (617) 258-1787.

[39] National Geophysical Data Center Solar-Terrestrial Physics Division FTP site. ftp://ftp.ngdc.noaa.gov/STP/GEOMAGNETIC_DATA/INDICES/KP_AP.

[40] Schatten, K. National Science Foundation. Personal correspondence with J. Fischer, Charles Stark Draper Laboratory. June, 1997.

[41] Wall, L, T. Christiansen, and R. Schwartz. *Programming Perl*. Sebastopol, CA: O'Reilly & Associates, 1996.

[42] Kelso, T.S. "Frequently Asked Questions: Two-Line Element Set Format." http://celestrak.com/columns/v04n03/index.html. Satellite Times. Accessed January, 1999.

[43] Koorts, Willie. Personal Home Page. http://canopus.saao.ac.za/~wpk/. Accessed February, 2000.

[44] McCants, Mike. Personal Home Page. http://www.fc.net:80/~mikem/. Accessed February, 2000.

[45] Carter, S.S. *Precision Orbit Determination From GPS Receiver Navigation Solutions*. CSDL-T-1260, Master of Science Thesis, Massachusetts Institute of Technology. Cambridge, MA, June 1996.

[46] Justus, C. et al. *The NASA/MSFC Global Reference Atmospheric Model – MOD3 (With Sperical Harmonic Wind Model)*. NASA Contractor Report 3256. NASA Contract NAS 8-32897, March, 1980.

[47] *The Upper Atmosphere of the Earth – Model of Density for Ballistic Maintenance of the Earth Artificial Satellite Flights.* GOST 25645.115-84. Moscow: State Committee of the USSR for Quality Control of Production and Standards, 1984.

[48] Laneve, G. "Small Satellites for Aeronomic Missions in the Lower Thermosphere." Paper AAS 97-729, Presented at AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, ID, August 1997.

[49] Thompsen, Allen. FTP Site for NORAD Two-Line Elements. ftp://kilroy.jpl.nasa.gov/pub/space/elements/satelem. Accessed February, 2000.

[50] Vallado, D. A. *Fundamentals of Astrodynamics and Applications.* Space Technology Series. New York: McGraw-Hill, 1997.

[51] King-Hele, D. *Theory of Satellite Orbits in an Atmosphere.* London: Butterworths, 1964.

[52] Cefola, P.J., D.J. Fonte and N. Shah. "The Inclusion of the Naval Space Command Satellite Theory PPT2 in the R&D GTDS Orbit Determination System." Paper AAS 96-142, *Advances in the Astronautical Sciences: Spaceflight Mechanics 1996*, Vol. 93, Part I. San Diego: Univelt, Inc, 1996, pp. 665-691.

[53] Nostrand, P.M. *Forecast Verification of the 10.7 Centimeter Solar Flux and the Ap Daily Geomagnetic Activity Indices.* Master of Science Thesis, Air Force Institute of Technology. Wright-Patterson Air Force Base, OH, December 1984.

[54] Cederqvist, Per et. al. *Version Management with CVS.* Signum Support AB: 1992.

[55] Hoots, F.R. and R.L. Roehrich. *Models for Propagation of NORAD Element Sets.* Spacetrack Report No. 3, Aerospace Defense Command, United States Air Force, December 1980.

[56] Mugellesi, R. and D.J. Kerridge. "Prediction of Solar and Geomagnetic Activity for Low-Flying Spacecraft." *European Space Agency (ESA) Journal*, Vol. 15, 1991, pp. 123-134.

[57] MATLAB Computer Software. Version 5.3.0.10183 (R11). Copyright 1984-1999, The Mathworks, Inc. Jan 1999.

[58] Clark, T.D.G, A.W.P Thomson, and D.J. Kerridge. *A Review of Methods of Forecasting Solar and Geomagnetic Activity in the Short-Term.* British Geological Survey Technical Report WM/91/23C, ESOC Contract Number 7558/88/D/IM(SC). Edinburgh: NERC, 1991.

[59] Gropp, William, and Ewing Lusk. *User's Guide for mpich, a Portable Implementation of MPI.* Mathematics and Computer Science Division, Argonne National Laboratory, ANL-96/6, 1996.

[60] Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing.* MIT Press, Cambridge, MA, 1994.